S01P0217US00

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年 ２月２１日

出 願 番 号
Application Number: 特願２０００－０４３３９０

出 願 人
Applicant (s): ソニー株式会社

CERTIFIED COPY OF PRIORITY DOCUMENT

２００１年 １月 ５日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号 出証特２０００－３１０９３９７

【書類名】　特許願
【整理番号】　00000202
【提出日】　平成12年 2月21日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 15/00
【発明の名称】　データ通信方法及びデータ通信装置、並びにプログラム記憶媒体
【請求項の数】　23
【発明者】
【住所又は居所】　東京都品川区北品川６丁目７番３５号　ソニー株式会社内
【氏名】　蒲生　勉
【特許出願人】
【識別番号】　000002185
【氏名又は名称】　ソニー株式会社
【代表者】　出井　伸之
【代理人】
【識別番号】　100101801
【弁理士】
【氏名又は名称】　山田　英治
【電話番号】　03-5541-7577
【選任した代理人】
【識別番号】　100093241
【弁理士】
【氏名又は名称】　宮田　正昭
【電話番号】　03-5541-7577
【選任した代理人】
【識別番号】　100086531
【弁理士】

【氏名又は名称】　澤田　俊夫

【電話番号】　03-5541-7577

【手数料の表示】

【予納台帳番号】　062721

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9904833

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データ通信方法及びデータ通信装置、並びにプログラム記憶媒体

【特許請求の範囲】

【請求項１】

メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信方法であって、

（ａ）該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップと、

（ｂ）該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信するステップと、

を具備することを特徴とするデータ通信方法。

【請求項２】

前記複合オブジェクトは、コンテキスト･スイッチが発生しない関数呼び出し形式で呼び出し可能な複数のオブジェクトで構成されることを特徴とする請求項１に記載のデータ通信方法。

【請求項３】

前記複合オブジェクト内の各オブジェクトは、以下の制約条件の少なくとも１つを満たすことを特徴とする請求項１に記載のデータ通信方法。

（１）各オブジェクト間に実行逐次性があることこと。

（２）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であること。

（３）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側にお

けるオブジェクトのそれと同等か又はそれ以上であること。

（４）各オブジェクトが同じメモリ保護属性を持つこと。

【請求項４】

さらに、

（ｃ）該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成するステップ、

を具備し、

前記ステップ（ｂ）では、該メッセージ通信履歴を基に該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したか否かを判別する、

ことを特徴とする請求項１に記載のデータ通信方法。

【請求項５】

さらに、

（ｃ）該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成するステップ、

を具備し、

前記ステップ（ｂ）では、前記複合オブジェクト内のオブジェクト実行終了時に該メッセージ通信履歴が異なる実行スレッドからの通信状態を示している場合に、蓄積されたメッセージを１回のメッセージ通信で送信先オブジェクトに送信する、

ことを特徴とする請求項１に記載のデータ通信方法。

【請求項６】

前記ステップ（ａ）では、該複合オブジェクトと該独立オブジェクトの状態の関係に応じてメッセージ通信の蓄積を制御することを特徴とする請求項１に記載のデータ通信方法。

【請求項７】

前記ステップ（ａ）では、該独立オブジェクトの状態に応じてメッセージ通信の蓄積を制御することを特徴とする請求項１に記載のデータ通信方法。

【請求項８】

該複合オブジェクト内のオブジェクトのメッセージ送信先となる該複合オブジ

ェクト外の独立オブジェクトが複数存在する場合、前記ステップ（ａ）では、メッセージ送信先毎にメッセージ通信の蓄積を制御することを特徴とする請求項１に記載のデータ通信方法。

【請求項９】

さらに、

（ｄ）送信元となる複合オブジェクト内のオブジェクトと送信先となる外部オブジェクトそれぞれの実行スレッドのスケジューリング優先度と割り込み優先度の関係に応じて、メッセージの蓄積を行うか又は即座に送信するかを決定するステップ、

を具備することを特徴とする請求項１に記載のデータ通信方法。

【請求項１０】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成されたオブジェクト指向オペレーティング･システムであることを特徴とする請求項１に記載のデータ通信方法。

【請求項１１】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成されたアプリケーション又はデバイス･ドライバであることを特徴とする請求項１に記載のデータ通信方法。

【請求項１２】

メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信装置であって、

（ａ）該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積する手段と、

（ｂ）該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信する手段と、

を具備することを特徴とするデータ通信装置。

【請求項１３】

前記複合オブジェクトは、コンテキスト･スイッチが発生しない関数呼び出し形式で呼び出し可能な複数のオブジェクトで構成されることを特徴とする請求項１２に記載のデータ通信装置。

【請求項１４】

前記複合オブジェクト内の各オブジェクトは、以下の制約条件の少なくとも１つを満たすことを特徴とする請求項１２に記載のデータ通信装置。

（１）各オブジェクト間に実行逐次性があることこと。

（２）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であること。

（３）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側におけるオブジェクトのそれと同等か又はそれ以上であること。

（４）各オブジェクトが同じメモリ保護属性を持つこと。

【請求項１５】

さらに、

（ｃ）該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成する手段、

を具備し、

前記手段（ｂ）は、該メッセージ通信履歴を基に該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したか否かを判別する、

ことを特徴とする請求項１２に記載のデータ通信装置。

【請求項１６】

さらに、

（ｃ）該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成する手段、

を具備し、

前記手段（ｂ）は、前記複合オブジェクト内のオブジェクト実行終了時に該メッセージ通信履歴が異なる実行スレッドからの通信状態を示している場合に、蓄積されたメッセージを１回のメッセージ通信で送信先オブジェクトに送信する、ことを特徴とする請求項１２に記載のデータ通信装置。

【請求項１７】

前記手段（ａ）は、該複合オブジェクトと該独立オブジェクトの状態の関係に応じてメッセージ通信の蓄積を制御することを特徴とする請求項１２に記載のデータ通信装置。

【請求項１８】

前記手段（ａ）では、該独立オブジェクトの状態に応じてメッセージ通信の蓄積を制御することを特徴とする請求項１２に記載のデータ通信装置。

【請求項１９】

該複合オブジェクト内のオブジェクトのメッセージ送信先となる該複合オブジェクト外の独立オブジェクトが複数存在する場合、前記手段（ａ）では、メッセージ送信先毎にメッセージ通信の蓄積を制御することを特徴とする請求項１２に記載のデータ通信装置。

【請求項２０】

さらに、

（ｄ）送信元となる複合オブジェクト内のオブジェクトと送信先となる外部オブジェクトそれぞれの実行スレッドのスケジューリング優先度と割り込み優先度の関係に応じて、メッセージの蓄積を行うか又は即座に送信するかを決定する手段

、

を具備することを特徴とする請求項１２に記載のデータ通信装置。

【請求項２１】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成されたオブジェクト指向オペレーティング・システムであることを特徴とする請求項１２に記載のデータ通信装置。

【請求項２２】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成

されたアプリケーション又はデバイス･ドライバであることを特徴とする請求項１２に記載のデータ通信装置。

【請求項２３】

メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信処理をコンピュータ･システム上で実行するためのコンピュータ･プログラムをコンピュータ可読形式で記憶するプログラム記憶媒体であって、前記コンピュータ･プログラムは、

（ａ）該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップと、

（ｂ）該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信するステップと、

を具備することを特徴とするプログラム記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ソフトウェア･プログラム間におけるデータ通信に係り、特に、複数のオブジェクト間で行われるデータ通信に関する。

【０００２】

更に詳しくは、本発明は、複数の並列オブジェクトで構成されるオブジェクト指向オペレーティング･システムにおいてオブジェクト間で行われるデータ通信に係り、特に、コンテキスト･スイッチなどのコスト発生を可能な限り抑制したオブジェクト間の高速なデータ通信に関する。

【０００３】

【従来の技術】

昨今のＬＳＩ技術の革新に伴い、情報処理機器、情報通信機器、家電機器など各種の機器は電子化され、マイクロプロセッサ上でソフトウェアを実行すること

によって機器動作の制御を行うことが一般的となってきた。

【０００４】

また、旧来は、制御コードを始めとする各種ソフトウェア･プログラムは、機器に内蔵されるＲＯＭ（Read Only Memory）上に恒久的に書き込んでおくという形態で、出荷前にあらかじめ機器内に導入することが一般的であった。しかし、最近では、機器内に書き込み可能な記憶装置が装備され、各種記憶媒体を介して最新のソフトウェアを機器の出荷後に導入することも可能となってきた。さらに、多くの機器は、ＬＡＮやインターネット、あるいは一般公衆回線などの通信媒体に接続されており、記憶媒体の物理的な流通に頼ることなく、最新のソフトウェアにリモート･アクセスすることが可能となってきている。

【０００５】

機器制御用のソフトウェアは、通常、オペレーティング･システム（ＯＳ）と呼ばれる形式で提供される。ＯＳに関する厳格な定義はないが、機器ハードウェアと上位アプリケーションとの仲立ち（すなわち、アプリケーションに対するＡＰＩ（Application Programming Interface）の提供）と、機器ハードウェアの統括的な制御を、その基本的な役割とする。機器とＯＳとの組み合わせは、一般に「プラットフォーム」と呼ばれる。オペレーティング･システムは、アプリケーションやデバイス・ドライバに対してその実行環境を提供する。また、ＯＳの介在により、異なる機種間で同一のアプリケーションを利用することが可能となる。

【０００６】

ところで、ソフトウェア開発の分野では、処理手順ではなく処理対象となるデータを重要視するという「オブジェクト指向」（object oriented）技術が盛んに採り入れられている。一般に、オブジェクト指向によれば、ソフトウェアの開発と保守が効率化されると考えられている。オブジェクト指向に基づくソフトウェアは、基本的に、データとそのデータに対する処理手続きとを一体化させた「オブジェクト」というモジュール単位で扱われる。また、必要に応じて複数のオブジェクトを作成したり組み合わせることで１つのソフトウェアが完成する。

【０００７】

オブジェクト指向パラダイムは、「カプセル化」、「クラス／インスタンス」、「クラス継承（インヘリタンス）」、及び、「メッセージ･パッシング」という４つの基本技術によって実現される。

【０００８】

カプセル化は、データと手続き（メソッド）を一体化させることを指す。クラスは、複数のオブジェクトを共通化して定義することを指す。インスタンスはクラスに属するオブジェクトの実体を表す（同一クラスに属するオブジェクトすなわちインスタンスは基本的に同一のメソッドを有するので、個々にメソッドを定義する必要がない）。インヘリタンスとは、あるクラスで定義したことを別の（例えば下位の）クラスが継承することを意味する（新たにクラスを定義するときには、定義済みのクラスとの差分のみを追加変更するだけでよい）。メッセージ･パッシングとは、オブジェクトにメッセージを送って所定の動作を指示することを言う。各オブジェクトは自身のデータを隠蔽しているので、通常、メッセージ･パッシング以外の方法でオブジェクトにアクセスすることはできない。

【０００９】

最近では、オペレーティング･システムに対しても、オブジェクト指向技術の適用が行われるようになってきている。すなわち、オペレーティング･システムの個々の構成要素を、実行主体である並行オブジェクトとしてモジュール化する。本明細書中では、以下、このようなオペレーティング･システムのことを「オブジェクト指向オペレーティング･システム」と呼び、また、並行オブジェクトのことを単に「オブジェクト」と呼ぶことにする。なお、オペレーティング･システムの構成要素となるオブジェクトと、オペレーティング･システム上で実行されるアプリケーションのオブジェクトが同様の実行機能を持つようなオブジェクト指向オペレーティング･システムのことを、特に、「純オブジェクト指向オペレーティング･システム」と称することもある。

【００１０】

オブジェクト指向オペレーティング･システムでは、オペレーティング･システムが提供する各々のサービスはオブジェクトの集合として定義される。この特性を活かす帰結として、オブジェクト指向オペレーティング･システムは、システ

ム構成の柔軟性や、システムの動的変更の容易性などの点で、旧来のオペレーティング・システムよりも優れた性能を確保することができる。

【００１１】

例えば、あるサービスを提供するためのオブジェクトを、オペレーティング・システムの実行開始前に適切に組み合わせることによって、それぞれのユーザにとって必要な機能を備えたオペレーティング・システムを容易に構築することができる。また、システムの最適化やアップグレードなどのためになされる機能追加や削除を、システム自体を停止させることなく動的に実施することも可能である。

【００１２】

オブジェクト指向オペレーティング・システムでは、システムのサービスを提供する各オブジェクトが互いに並行して動作する。各オブジェクトはモジュール化されているので、メッセージ通信機構を用いることで、オブジェクト同士で通信してメッセージの交換を行ったり、動作の同期をとることができる。

【００１３】

オブジェクト間のメッセージ通信には、オブジェクトの実行スレッドの状態の待避とその復元に伴うコスト、メッセージの配送のコストなどを費やしてしまう。このため、オブジェクト指向オペレーティング・システムの場合、オブジェクト間のメッセージ通信が頻繁に生じると、その回数だけコンテキスト・スイッチが発生してしまうので、システム全体の性能が劣化してしまうという問題がある。

#d

したがって、オブジェクト指向オペレーティング・システムのシステム設計者においては、システム構成の柔軟性やシステムの動的変更の容易性などオブジェクト指向オペレーティング・システムの優れた特徴を採り入れるだけでなく、かかる特徴と実際の実行性能とのバランスを充分に考慮した上でオブジェクトを設計する必要がある。

【００１４】

例えば、本出願人に既に譲渡されている特願平１１－５７６８９号明細書には

、オブジェクト間のメッセージ通信時のコストを削減することでシステム全体の実行性能を向上させることができるデータ処理方法について開示されている。同明細書に記載のデータ処理方法では、メッセージ通信が頻繁に発生するような複数のオブジェクトを包括するようなオブジェクト、すなわち「複合オブジェクト」を定義するようになっている。

【００１５】

複合オブジェクトを構成する各オブジェクトのプログラミング･スタイルは、複合オブジェクト外の通常の並行オブジェクトのそれと同様であり（すなわち変更を要しない）、オブジェクト識別子（Object Identifier：ＯＩＤ）を有して、通常のオブジェクトからも参照可能である。また、複合オブジェクト内の各オブジェクトは、独自の実行スレッドを持たず、複合オブジェクトが持つ実行スレッドを共有するようになっている。そして、同一の複合オブジェクト内でオブジェクト間のメッセージ通信を行う時には、実行スレッドの切り替えすなわちコンテキスト･スイッチが発生しないので、メッセージ通信のコストを関数呼び出しと同等にすることができる。

【００１６】

オブジェクト指向オペレーティング･システム上で複合オブジェクトを構成することにより、システムの柔軟性を維持したまま、メッセージ通信のコストを削減することができるので、同特許出願に係るデータ処理方法は非常に有効と言えよう。

【００１７】

ここで、２以上のオブジェクトを包括して複合オブジェクトを構成する場合には、元の各オブジェクトに対しては所定の制約条件が課される。この制約条件は、オペレーティング･システムが提供する機能によって若干の相違はあるが、おおむね以下の通りである。

【００１８】

（１）各オブジェクト間に実行逐次性があること。より具体的には、以下の２つの条件を満たす場合に「実行逐次性」が認められる。すなわち、

（１－１）一方のオブジェクトから他方のオブジェクトにメッセージを送信する

とき、両オブジェクトが並列動作する必要がない。

（１－２）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側のオブジェクトが他のメッセージを処理中であることがない。

【００１９】

（２）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であること。

【００２０】

（３）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側におけるオブジェクトのそれと同等か又はそれ以上であること。

【００２１】

（４）各オブジェクトが同じメモリ保護属性を持つこと。

【００２２】

言い換えれば、上記の制約条件を満足しないようなオブジェクト同士を構成オブジェクトとして複合オブジェクトを構成することはできない。特願平１１－５７６８９号明細書は、複合オブジェクトを構成するための手順について詳解している。但し、すべてのオブジェクトを複合オブジェクトとして包括できるものではない。

【００２３】

このため、頻繁にメッセージ通信を行うような特定のオブジェクトの組み合わせがあったとしても、上記の制約条件を満足しない限り、複合オブジェクトに包括せず、通常の並行オブジェクトとして独立させる必要がある。このため、頻繁に行われるメッセージ通信のために、実行スレッドの切り替えすなわちコンテキスト･スイッチが発生してしまい、メッセージ通信のコストを削減することができない。

【００２４】

例えば、タスクのスケジューリングを管理するスケジューラのようなオブジェクトは、特定の複合オブジェクトの内部に包括することはできない。したがって、複合オブジェクト内でメッセージ通信を行う場合であっても、各構成オブジェクトがスケジューラをコールする度にコンテキストが切り替わり、コストが増大してしまう。スケジューラのコールは、データ処理の過程において、極めて頻繁に発生する。

【００２５】

【発明が解決しようとする課題】

本発明の目的は、複数のオブジェクト間でメッセージ交換を行うための、優れたデータ通信方式を提供することにある。

【００２６】

本発明の更なる目的は、複数の並列オブジェクトで構成されるオブジェクト指向オペレーティング・システムにおいてオブジェクト間でのメッセージ交換を好適に行うことができるの、優れたデータ通信方式を提供することにある。

【００２７】

本発明の更なる目的は、コンテキスト・スイッチなどのコスト発生を可能な限り抑制して、オブジェクト間のメッセージ交換を高速化することができる、優れたデータ通信方式を提供することにある。

【００２８】

本発明の更なる目的は、オブジェクト指向オペレーティング・システムにおけるシステム構成の柔軟性や動的変更の容易性などの優位性を維持したままで、オブジェクト間のメッセージ通信時のコストを削減して、システム全体の実行性能を向上させることにある。

【００２９】

本発明の更なる目的は、複合オブジェクト内の各構成オブジェクトが外部の特定のオブジェクトとの間で頻繁にメッセージ通信を行う場合であっても、各オブジェクトの独立性を保ちつつ、コンテキスト・スイッチに伴うコストを削減して、システム全体の実行性能を向上させることにある。

【００３０】

【課題を解決するための手段及び作用】

本発明は、上記課題を参酌してなされたものであり、その第１の側面は、メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信方法又はデータ通信装置であって、

（ａ）該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップ又は手段と、

（ｂ）該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して一時蓄積されたメッセージを該独立オブジェクトに対して一時に送信するステップ又は手段と、

を具備することを特徴とするデータ通信方法又はデータ通信装置である。

【００３１】

本発明の第１の側面に係るデータ通信方法又はデータ通信装置は、例えば、メッセージ通信を行う複数の並行オブジェクトからなるオブジェクト指向オペレーティング･システムが導入されたデータ処理システム上で実現される。

【００３２】

該データ通信方法及びデータ通信装置によれば、まず、メッセージ通信が頻繁に発生するような複数のオブジェクトを包括した複合オブジェクトが生成される。複合オブジェクトは実行逐次型のオブジェクトで構成され、複合オブジェクト内でのオブジェクト間通信は関数呼び出しで実現するので、メッセージ通信に伴うコンテキスト･スイッチすなわちコストは発生しない。したがって、同一の複合オブジェクト内のオブジェクトがコールされたときには、同一の実行スレッドによってメッセージ通信を実行することができる。

【００３３】

他方、複合オブジェクト内から外部の独立オブジェクトに対してメッセージ通信を行うと、コンテキストが切り替わる。複合オブジェクト内においてオブジェクト間通信を行う場合であっても、スケジューラなどの複合オブジェクト外に存在する独立オブジェクトを頻繁にコールしなければならないことがある。コール

の度にメッセージ通信を実行していると、コンテキスト･スイッチの発生によりコストが著しく増大してしまう。

【００３４】

そこで、本発明の第１の側面に係るデータ通信方法及びデータ通信装置では、複合オブジェクト内のオブジェクトが外部の独立オブジェクトをコールしたときには、逐次的にメッセージ通信を実行する代わりに、独立オブジェクトへのメッセージ通信をキュー内に一時的に蓄積しておく。そして、複合オブジェクトと独立オブジェクトの関係が所定の状態に到達したときに、独立オブジェクトへのメッセージを一括して送信するようにした。

【００３５】

この結果、コンテキスト･スイッチなどのコスト発生を可能な限り抑制して、オブジェクト間のメッセージ交換を高速化することができる。

【００３６】

また、本発明によれば、オブジェクト指向オペレーティング･システムにおけるシステム構成の柔軟性や動的変更の容易性などの優位性を維持したままで、オブジェクト間のメッセージ通信時のコストを削減して、システム全体の実行性能を向上させることができる。

【００３７】

また、本発明によれば、複合オブジェクト内の各構成オブジェクトが外部の特定のオブジェクトとの間で頻繁にメッセージ通信を行う場合であっても、各オブジェクトの独立性を保ちつつ、コンテキスト･スイッチに伴うコストを削減して、システム全体の実行性能を向上させることができる。

【００３８】

ここで、複合オブジェクト内の各オブジェクトは、例えば以下に示すような制約条件を満足している。すなわち、

（１）各オブジェクト間に実行逐次性があることこと。

（２）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であ

ること。

（３）一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側におけるオブジェクトのそれと同等か又はそれ以上であること。

（４）各オブジェクトが同じメモリ保護属性を持つこと。

【００３９】

また、本発明の第１の側面に係るデータ通信方法又はデータ通信装置は、さらに、（ｃ）該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成するステップ又は手段を具備してもよい。このような場合、前記ステップ又は手段（ｂ）では、該メッセージ通信履歴を基に該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したか否かを判別することができる。あるいは、前記ステップ又は手段（ｂ）では、前記複合オブジェクト内のオブジェクト実行終了時に該メッセージ通信履歴が異なる実行スレッドからの通信状態を示している場合には、蓄積されたメッセージを１回のメッセージ通信で送信先オブジェクトに送信することができる。

【００４０】

また、前記ステップ又は手段（ａ）では、該複合オブジェクトと該独立オブジェクトの状態の関係に応じてメッセージ通信の蓄積を制御するようにしてもよい。

【００４１】

あるいは、前記ステップ又は手段（ａ）では、該独立オブジェクトの状態に応じてメッセージ通信の蓄積を制御するようにしてもよい。

【００４２】

例えば、複合オブジェクト内でのオブジェクト間通信、すなわち複合オブジェクト内のあるオブジェクトによるメッセージ送信先が同一複合オブジェクト内の他のオブジェクトである場合には、同一の実行スレッドでメッセージ通信を実行する。このとき、メッセージ通信履歴情報を同一実行スレッドからの通信状態に更新すればよい。

【００４３】

これに対し、複合オブジェクト内のオブジェクトのメッセージ送信先が複合オブジェクト外に存在する通常のオブジェクトである場合には、即座にメッセージ通信を行わずに、メッセージ送信先に応じてメッセージを一旦蓄積する。そして、複合オブジェクト内でのメッセージ処理が終了すると、メッセージ通信履歴情報を参照する。メッセージ通信履歴情報が同一実行スレッドからの通信状態を示しているときは復帰処理を行う。他方、メッセージ通信履歴が異なる実行スレッドからの通信状態を示しているときには、蓄積されたメッセージを１回のメッセージ通信で送信先オブジェクトに送信して、複合オブジェクトとしてのメソッドの実行を完了させる。

【００４４】

また、該複合オブジェクト内のオブジェクトのメッセージ送信先となる該複合オブジェクト外の独立オブジェクトが複数存在する場合、前記ステップ又は手段（ａ）では、メッセージ送信先毎にメッセージ通信の蓄積を制御するようにしてもよい。

【００４５】

また、送信元となる複合オブジェクト内のオブジェクトと送信先となる外部オブジェクトそれぞれの実行スレッドのスケジューリング優先度と割り込み優先度の関係に応じて、メッセージの蓄積を行うか又は即座に送信するかを決定するようにしてもよい。

【００４６】

本発明の第１の側面に係るデータ通信方法によれば、複合オブジェクト内のオブジェクトと複合オブジェクト外に存在するオブジェクトとのメッセージ通信が頻繁に行われている場合、複合オブジェクト内の各オブジェクトには透過的な形式でメッセージが蓄積され、その後の適切なタイミングで蓄積された複数のメッセージをまとめて１回のメッセージ通信による送信することができる。この結果、メッセージ通信のコストを大幅に削減することができる。また、オブジェクトの独立性を保持したままシステム全体の実行性能を向上させることができる。

【００４７】

本発明の第１の側面に係るデータ通信方法は、オブジェクト指向オペレーティ

ング･システムだけではなく、複数の並行オブジェクトで構成されたアプリケーションやデバイス･ドライバに対しても適用することができる。

【００４８】

また、本発明の第２の側面は、メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信処理をコンピュータ･システム上で実行するためのコンピュータ･プログラムをコンピュータ可読形式で記憶するプログラム記憶媒体であって、前記コンピュータ･プログラムは、

（ａ）該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップと、

（ｂ）該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信するステップと、

を具備することを特徴とするプログラム記憶媒体である。

【００４９】

本発明の第２の側面に係るコンピュータ可読記憶媒体は、例えば、様々なプログラム･コードを実行可能な汎用コンピュータ･システムに対して、コンピュータ・プログラムをコンピュータ可読な形式で提供する媒体である。このような媒体は、例えば、ＣＤ（Compact Disc）やＦＤ（Floppy Disc）、ＭＯ（Magneto-Optical disc）などの着脱自在で可搬性の記憶媒体である。あるいは、ネットワーク（ネットワークは無線、有線の区別を問わない）などの伝送媒体などを経由してコンピュータ・プログラムを特定のコンピュータ・システムに提供することも技術的に可能である。

【００５０】

このようなプログラム記憶媒体は、コンピュータ・システム上で所定のコンピュータ・プログラムの機能を実現するための、コンピュータ・プログラムと記憶媒体との構造上又は機能上の協働的関係を定義したものである。換言すれば、本発明の第２の側面に係るプログラム記憶媒体を介して所定のコンピュータ・プロ

グラムをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の第１の側面と同様の作用効果を得ることができる。

【００５１】

本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【００５２】

【発明の実施の形態】

以下、図面を参照しながら本発明の実施例を詳解する。

【００５３】

図１には、本発明の実施に供されるデータ処理システム１０のハードウェア構成を模式的に示している。データ処理システム１０の一例は、米ＩＢＭ社のパーソナル･コンピュータＰＣ／ＡＴ（Personal Computer/Advanced Technology）の互換機又は後継機を始めとする情報処理機器であるが、その他のタイプの情報処理機器や情報家電機器であってもよい。以下、システム１０の各部について説明する。

【００５４】

システム１０のメイン・コントローラであるプロセッサ１１は、例えば、ＣＰＵ（Central Processing Unit）と呼ばれるＬＳＩ（Large Scale Integration）チップで構成され、オペレーティング･システムの制御下で、各種のアプリケーションを実行するようになっている。

【００５５】

図示の通り、プロセッサ１１は、バス１７によって他の機器類と相互接続されている。バス１７上の各機器にはそれぞれ固有のメモリ･アドレス又はＩ／Ｏアドレスが付与されており、プロセッサ１１はこれらアドレスによって機器アクセスが可能となっている。バス１７の一例はＰＣＩ（Peripheral Component Interconnect）バスである。

【００５６】

メモリ１２は、プロセッサ１１において実行されるプログラム･コードを格納

したり、実行中の作業データを一時保管するために使用される記憶装置である。本実施例では、メモリ１２は、不揮発及び揮発メモリ双方を含むものと理解されたい。

【００５７】

ディスプレイ・コントローラ１３は、プロセッサ１１が発行する描画命令を実際に処理するための専用コントローラであり、例えばＳＶＧＡ（Super Video Graphic Array）又はＸＧＡ（eXtended Graphic Array）相当の描画機能をサポートする。ディスプレイ･コントローラ１３において処理された描画データは、例えばフレーム・バッファ（図示しない）に一旦書き込まれた後、表示装置２１に画面出力される。表示装置２１は、例えば、ＣＲＴ（Cathode Ray Tube）ディスプレイや、液晶表示ディスプレイ（Liquid Crystal Display）などである。

【００５８】

入力機器インターフェース１４は、キーボード２２やマウス２３などのユーザ入力機器をシステム１０に接続するための装置である。入力機器インターフェース１４は、キーボード２２によるキー入力又はマウス２３を介した座標指示入力に応答して、プロセッサ１１に対して割り込みを発生する。

【００５９】

ネットワーク･インターフェース１５は、Ｅｔｈｅｒｎｅｔなどの所定の通信プロトコルに従って、システム１０をＬＡＮ（Local Area Network）などのネットワークに接続することができる。ネットワーク･インターフェース１５は、一般に、ＬＡＮアダプタ・カードの形態で提供され、マザーボード（図示しない）上のＰＣＩバス・スロットの装着して用いられる。

【００６０】

ＬＡＮ上には、複数のホスト（コンピュータ）がトランスペアレントに接続され、分散コンピューティング環境が構築されている。また、ホストの一部はルータとして稼動し、さらに他のＬＡＮやインターネットなどの外部ネットワークに接続されている。インターネット上では、ソフトウェア・プログラムやデータ・コンテンツなどのディストリビューションが行われる（周知）。

【００６１】

ＨＤＤ（Hard Disc Drive）インターフェース１６は、ハード･ディスク･ドライブなどの外部記憶装置をシステム１０に接続するための装置である。ＨＤＤインターフェース１６は、例えば、ＩＤＥ（Integrated Drive Electronics）やＳＣＳＩ（Small Computer System Interface）などのインターフェース規格に準拠する。

【００６２】

ＨＤＤ２４は、記憶担体としての磁気ディスクを固定的に搭載した外部記憶装置であり（周知）、記憶容量やデータ転送速度などの点で他の外部記憶装置よりも優れている。ソフトウェア・プログラムを実行可能な状態でＨＤＤ２６上に置くことをプログラムのシステムへの「インストール」と呼ぶ。通常、ＨＤＤ２４には、プロセッサ１１が実行すべきオペレーティング･システムのプログラム･コードや、アプリケーション･プログラム、デバイス･ドライバなどが不揮発的に格納されている。

【００６３】

本実施例で言うオペレーティング･システムは、個々の構成要素が実行主体としての並列オブジェクトとしてモジュール化されて構成された「オブジェクト指向オペレーティング･システム」である。オブジェクト指向に基づくソフトウェアは、基本的に、データとそのデータに対する処理手続きとを一体化させた「オブジェクト」というモジュール単位で扱われる。オブジェクト指向プログラミングによれば、必要に応じて複数のオブジェクトを作成したり組み合わせることで１つのソフトウェアが完成する。

【００６４】

また、本実施例では、アプリケーションやデバイス･ドライバなどのオブジェクトで構成されるものとする。さらに、アプリケーション用の実行環境（以下、本明細書では”ｍＣＯＯＰ”と呼ぶことにする）と、デバイス･ドライバ用の実行環境（以下、本明細書では”ｍＤｒｉｖｅ”と呼ぶことにする）はともに、オペレーティング･システムによって提供されているものとする。例えば、表示装置２１の画面上に動画像を表示するためのアプリケーションや、画面表示を介した対話的なユーザ入力を実現するグラフィカル･ユーザ･インターフェース（ＧＵＩ）

を用いたプログラムなどは、ｍＣＯＯＰ上で実行される。また、ディスプレイ・コントローラ１３、キーボード２２、マウス２３、ＨＤＤインターフェース１６などのハードウェアすなわちデバイスを直接操作する各デバイス・ドライバは、ｍＤｒｉｖｅ上で実行される。

【００６５】

このようなシステム１０において、ｍＣＯＯＰ上で動作するアプリケーションとｍＤｒｉｖｅ上で動作するドライバに対しては、同一実行環境上のプログラム間だけでなく、異なる実行環境上のプログラム間でもメッセージ通信を行うことができるようになっている。例えば、ｍＣＯＯＰ上で対話的な画面入力を制御するプログラムは、ディスプレイ・コントローラ１３制御用のデバイス・ドライバに対してメッセージ通信を行うことでＧＵＩの実際の描画を実現するが、このデバイス・ドライバは異なる実行環境のｍＤｒｉｖｅ上で動作している。

【００６６】

次いで、本実施例に係るオブジェクト指向オペレーティング・システムが提供する実行環境について説明する。

【００６７】

図２には、本実施例に係るデータ処理システム１０が有するソフトウェア環境を模式的に示している。

【００６８】

同図において、オブジェクト２、オブジェクト３、及び、オブジェクト６は、オペレーティング・システムを構成するオブジェクトである。また、オブジェクト２とオブジェクト３は包括されて、１つの複合オブジェクト１を形成している。以下、オブジェクト２及び３を、複合オブジェクト１の「構成オブジェクト」と呼ぶことにする。

【００６９】

実行環境７は、複合オブジェクト１やオブジェクト６などオペレーティング・システムを構成する各オブジェクトに対して動作環境を提供する。また、複合オブジェクト１内では、構成オブジェクト２及び３の動作をサポートする独自の実行環境（以下、「構成オブジェクト実行環境」とする）４が用意されている。さ

らに、本実施例では、複合オブジェクト１内には、各構成オブジェクトのメッセージ通信履歴に関する情報を保持するコール･ヒストリ８と、構成オブジェクトが複合オブジェクト外のオブジェクトへ送出するメッセージ通信を格納するリクエスト･キュー９が配設されている。また、実行環境７上には、オブジェクト識別子ＯＩＤから参照されるオブジェクト記述情報を保持するオブジェクト･テーブル５が用意されている。

【００７０】

複合オブジェクト１の外部のオブジェクトからの複合オブジェクト１内の構成オブジェクト２又は３に対する実行要求は、構成オブジェクト実行環境４に向けて送信される。

【００７１】

構成オブジェクト実行環境４は、該実行要求に応答して、コール･ヒストリ８の状態を異なる実行スレッドからの通信状態に更新して、該当する構成オブジェクトを実行するようになっている。

【００７２】

逆に、複合オブジェクト１内の構成オブジェクトから複合オブジェクト１外のオブジェクトに対するメッセージ送信要求が発行されると、該要求は、同じ複合オブジェクト１内にある構成オブジェクト実行環境４に送られる。

【００７３】

構成オブジェクト実行環境４は、まず、メッセージ受信先のオブジェクトをオブジェクト･テーブル５内で検索して、該受信側オブジェクトが同一の複合オブジェクト１内に存在するか否かを判断する。

【００７４】

メッセージ受信側が同じ複合オブジェクト１内に存在する場合には、構成オブジェクト実行環境４は、コール･ヒストリ８の状態を同一実行スレッド内からの通信状態に更新して、同一実行スレッド内で該当する構成オブジェクトを実行する。

【００７５】

これに対し、メッセージ受信先が複合オブジェクト１の外部のオブジェクトで

あると判断された場合には、構成オブジェクト実行環境４は、コール･ヒストリ８の状態を異なる実行スレッドからの通信状態に更新するとともに、メッセージ受信先のオブジェクトを示すオブジェクト識別子を付けて、該メッセージをリクエスト･キュー９に一旦格納する。

【００７６】

複合オブジェクト１は、その実行が終了すると、実行終了要求を構成オブジェクト実行環境４に送る。構成オブジェクト実行環境４は、コール･ヒストリ８の状態を確認して、同一実行スレッドからの通信状態であった場合には、メッセージ送信元である構成オブジェクトの実行を同一スレッドで再開させる。再開させる時点で、コール･ヒストリ８を１つ前の状態に戻す。これに対し、異なる実行スレッドからの通信状態であることをコール･ヒストリ８が示す場合には、構成オブジェクト実行環境４は、リクエスト･キュー９に格納されている１以上のメッセージを１回のメッセージ通信でメッセージ送信先に送信して、通常のオブジェクトとしての実行を終了する。

【００７７】

上述したような構成を持つオブジェクト指向オペレーティング･システムによれば、同一複合オブジェクト内における構成オブジェクト間のメッセージ通信のコストが削減されるだけでなく、複合オブジェクト外にある通常のオブジェクトへのメッセージ通信が頻繁に行われる場合であっても、構成オブジェクトからは透過的にメッセージ通信コストを削減することが可能である、という点を充分理解されたい。

【００７８】

なお、図２に示すようなソフトウェア構成は、オペレーティング･システムだけではなく、アプリケーションやデバイス･ドライバに対しても同様に適用することができる。

【００７９】

次いで、本実施例に係るオブジェクト指向オペレーティング･システムが提供する異なる実行環境間におけるメッセージ通信サービスについて説明する。

【００８０】

本実施例に係るオブジェクト指向オペレーティング･システムでは、上述したように、異なる実行環境として、アプリケーション用の実行環境ｍＣＯＯＰと、デバイス･ドライバ用の実行環境ｍＤｒｉｖｅを同時に提供する。

【００８１】

図３には、実行環境ｍＣＯＯＰ上で動作するアプリケーション５１と実行環境ｍＤｒｉｖｅ上で動作するデバイス･ドライバ５２との間でメッセージ通信が行われる様子を示している。

【００８２】

アプリケーション５１は、ｍＣＯＯＰ上で動作するオブジェクトであり、オペレーティング･システム５３が提供するＡＰＩ（Application Programming Interface）を利用することができる。アプリケーション５１は、メッセージ送信を要求するときには、ｍＣＯＯＰで提供されているＡＰＩ”ＳｅｎｄＷｉｔｈＲＢｏｘ”を用いる。このとき、ＳｅｎｄＷｉｔｈＲＢｏｘへの引数としてメッセージ受信側のオブジェクトを指定するオブジェクト識別子ＯＩＤを付けて、メッセージを渡す。このときの送信先オブジェクトが、実行環境ｍＤｉｒｖｅ上で動作するデバイス･ドライバ５２であったとする。オペレーティング･システム５３は、メッセージに付されたＯＩＤで送信先オブジェクトを特定して、ドライバ５２にメッセージを送信するとともに（ｉｎｖｏｋｅ）、同時にオブジェクト５１の実行を再開させる（ｒｅｓｕｍｅ）。

【００８３】

図４には、メッセージ通信時のオペレーティング･システム５３内での動作を詳細に示している。以下、同図を参照しながら、各オブジェクトの相関について説明する。

【００８４】

以下では、オペレーティング･システムを構成するオブジェクトのことを「メタ･オブジェクト」と呼び、アプリケーションやドライブを構成するオブジェクトのことを「ペース･オブジェクト」と呼ぶ。

【００８５】

オペレーティング･システムを構成するメタ･オブジェクトに対しては、ｍＣＯ

ＯＰやｍＤｒｉｖｅなどの各ベース･オブジェクト用の実行環境とは相違する、オペレーティング･システム構成用の実行環境が提供されている。本明細書中では、オペレーティング･システム構成用の実行環境のことを、”ｍＣｏｒｅ”と呼ぶことにする。システム実行環境であるｍＣｏｒｅは、メタ･オブジェクトに対して、幾つかのＡＰＩ（Application Programming Interface）を提供している。

【００８６】

メタ･オブジェクトは２つの方法で起動される。１つは、メタ･オブジェクトに送信されるメッセージによって起動される場合であり、もう１つはベース・オブジェクトが発行するＡＰＩにより起動される場合である。本明細書では、後者のメタ･オブジェクト起動方法のことを特に「メタ･コール」と呼ぶことにする。ベース･オブジェクトの実行環境で提供されるメッセージ通信などのＡＰＩをベース･オブジェクトが使用すると、メタ･コールを発行する。メタ･コールによって、ベース･オブジェクトの実行スレッドから、メタ･オブジェクトの実行スレッドへの遷移が行われる。ベース･オブジェクトとメタ･オブジェクトの動作レベルを切り替えることで、オペレーティング･システムとアプリケーションやドライバとの間に境界を形成して、システム全体の安全性を高めることができる。

【００８７】

ここで、異なる実行環境間でメッセージ通信サービスを行う際に必要となるＡＰＩについて説明しておく。

【００８８】

Ｓｅｎｄ：

このＡＰＩは、送信メタ･オブジェクトから引数で指定される受信メタ･オブジェクトに対して、引数で指定されるメッセージを送信するためのＡＰＩであり、引数で指定されるメソッドを起動する。受信メタ･オブジェクトがメッセージを受信した後、送信メタ･オブジェクトと受信メタ･オブジェクトは並行に動作する。また、送信メタ･オブジェクトから受信メタ・オブジェクトにメッセージが送信されたときに、受信メタ･オブジェクトが別のメッセージを処理中の場合には、その処理が終了するまで、送信メッセージの処理は開始されず、該メッセージ

はメッセージ･キュー９に格納される。

【００８９】

Ｅｘｉｔ：

このＡＰＩは、上記の”Ｓｅｎｄ”ＡＰＩで起動されたメタ･オブジェクトの実行を終了する。このＡＰＩが発行された時点で、メッセージ･キュー９に格納されているメッセージがある場合には、そのメッセージを開始させる。

【００９０】

ＲｅｓｕｍｅＢａｓｅ：

このＡＰＩは、メタ･コールにより起動されたメタ･オブジェクトの実行を終了し、メタ･コールを発行したベース･オブジェクトの実行を再開する。これは、メタ･オブジェクトからスケジューラ（Ｓｃｈｅｄｕｌｅｒ）６６へのメタ･コールになる。

【００９１】

ＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌ：

このＡＰＩは、メタ･コールにより起動されたメタ･オブジェクトの実行を終了する。但し、メタ･コールを発行したベース･オブジェクトの実行は停止したままにする。これは、メタ･オブジェクトからスケジューラ６６へのメタ･コールになる。

【００９２】

なお、ｍＣｏｒｅでは、実際には、上述した以外にも数多のＡＰＩを提供されているが、本発明の要旨とは直接関連しないので本明細書中では説明を省略する。また、異なる実行環境間のメッセージ通信サービスを実現するために、本出願人に既に譲渡されている特願平１０－２８３２０５号明細書に記載されているタグを用いる方法を使用することができる。

【００９３】

図４中に示されるｍＣＯＯＰＭａｉｌｅｒ６４は、オペレーティング･システム５３を構成するオブジェクトの１つであり、アプリケーション実行環境であるｍＣＯＯＰに対してメッセージ通信サービスを提供する。本実施例では、ｍＣＯＯＰＭａｉｌｅｒ６４は、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７の

構成オブジェクトとして実現されている。

【００９４】

ｍＣＯＯＰＭａｉｌｅｒ６４は、以下に示すメッセージを受け取り、該当する各動作を実行するようになっている。

【００９５】

【表１】

メッセージ	アクション
SendWithRBox	1. RBoxを作成し、RIDを返す。 2. 受信オブジェクトが同一実行環境であれば、受信オブジェクトにメッセージを配送する。 3. 受信オブジェクトが異なる実行環境であれば、その実行環境でメッセージを実現するメタ･オブジェクトに対しDeliverオブジェクトを送信する。
Receive	1. RIDからRBoxを検索する。 2. RBoxに結果メッセージを配送する。
Reply	1. RIDをTIDに変換する。 2. TIDが同一実行環境のものであれば、RIDに対応するRBoxに結果メッセージを配送する。 3. TIDが異なる実行環境のものであれば、DeliverTagメッセージを送信する。
Exit	1. メッセージ･キューにメッセージがあれば、送信する。 2. そうでなければオブジェクトの実行を終了させる。
Deliver	メッセージを配送する。
DeliverTag	結果メッセージを配送する。

【００９６】

ｍＣＯＯＰＭａｉｌｅｒ６４は、ｍＣＯＯＰ上で動作中のベース･オブジェクト（例えば、図４中のアプリケーション５２）から送られてくるＳｅｎｄＷｉｔｈＲＢｏｘメッセージにより起動して、所定の手続きを実行するようになっている。

【００９７】

図５には、ｍＣＯＯＰＭａｉｌｅｒ６４がベース･オブジェクトからＳｅｎｄＷｉｔｈＲＢｏｘメッセージを受信したことに応答して実行する手続きをフローチャート形式で示している。以下、このフローチャートに従って説明する。

【００９８】

まず、ステップＳ１では、結果メッセージを格納するためのＲＢｏｘを作成する。そして、ＲＢｏｘを識別するための識別子ＲＩＤをメッセージ送信元のオブジェクトに返す。

【００９９】

次いで、ステップＳ２では、ｍＣＯＯＰＭａｉｌｅｒ６４は、メッセージ受信側のオブジェクトが同一の実行環境上に存在するか否かを検査する。

【０１００】

受信側オブジェクトが同一実行環境上にある場合には、ステップＳ３に進み、該受信側オブジェクトが別のメッセージを処理中か否かをさらに検査する。

【０１０１】

別のメッセージを処理中でないときには、ステップＳ４に進み、スケジューラ６６に対してＩｎｖｏｋｅメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【０１０２】

他方、受信側オブジェクトが別のメッセージを処理中である場合には、ステップＳ６に進み、メッセージ･キュー９にメッセージを格納する。

【０１０３】

また、受信オブジェクトが異なる実行環境上にある場合には、ステップＳ７に進み、その実行環境上でメッセージ通信を実現しているメタ･オブジェクトに対して、Ｄｅｌｉｖｅｒメッセージを送信する。このとき、ＳｅｎｄＷｉｔｈＲＢｏｘメッセージでベース･オブジェクトが指定したメッセージとＲＩＤをＴＩＤに変換したものも送信する。

【０１０４】

最後に、”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩによってベース･オブジェクトの実行を再開させる（ステップＳ５）。

【０１０５】

また。ｍＣＯＯＰＭａｉｌｅｒ６４は、ｍＣＯＯＰのベース･オブジェクトから送られてきたＲｅｃｅｉｖｅメッセージによって起動すると、引数で指定され

たＲＩＤに対応するＲＢｏｘを検索する。

【０１０６】

ＲＢｏｘに既に結果メッセージが格納されていれば、その結果メッセージをベース･オブジェクトに返し、ＡＰＩ”ＲｅｓｕｍｅＢａｓｅ”を発行して、ベース･オブジェクトの実行を再開させる。他方、ＲＢｏｘに未だ結果メッセージが格納されていなければ、結果メッセージの受け取り領域をＲＢｏｘに設定して、ＡＰＩ”ＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌ”を発行して、ベース･オブジェクトの実行を停止させたまま、ｍＣＯＯＰＭａｉｌｅｒ６４の実行を終了する。

【０１０７】

また、ｍＣＯＯＰＭａｉｌｅｒ６４は、ｍＣＯＯＰのベース･オブジェクトから送られるＲｅｐｌｙメッセージにより起動すると、ＡＰＩ”Ｒｅｐｌｙ”を発行した実行スレッドに対応するＲＩＤをＴＩＤに変換して、ＴＩＤが同一実行環境上の識別子であるか否かを検査する。

【０１０８】

ＴＩＤが同一実行環境上のものであった場合には、ＲＩＤによりＲＢｏｘを検索する。ＲＢｏｘに対して既に”Ｒｅｃｅｉｖｅ” ＡＰＩが発行されていれば、ＡＰＩ”Ｒｅｐｌｙ”の引数で指定された結果メッセージを”Ｒｅｃｅｉｖｅ” ＡＰＩを発行したベース･オブジェクトが指定した領域に格納する。そして、スケジューラ６６にＲｅｓｕｍｅメッセージを送信して、”Ｒｅｃｅｉｖｅ” ＡＰＩを発行したベース･オブジェクトの実行を再開させる。また、ＲＢｏｘに対して未だ”Ｒｅｃｅｉｖｅ” ＡＰＩが発行されていなければ、”Ｒｅｐｌｙ”ＡＰＩの引数で指定された結果メッセージをＲＢｏｘに格納する。

【０１０９】

他方、ＴＩＤが異なる実行環境上のものであった場合には、ＴＩＤから結果メッセージを配送する実行環境のメッセージ通信を実現しているメタ･オブジェクトを特定して、そのメタ･オブジェクトに対して、ＤｅｌｉｖｅｒＴａｇメッセージを送信する。このとき、結果メッセージとＴＩＤを指定する。最後に、”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩにより、アプリケーションの実行を再開させる。

【０１１０】

また、ｍＣＯＯＰＭａｉｌｅｒ６４は、ｍＣＯＯＰのベース･オブジェクトから送られるＥｘｉｔメッセージにより起動されると、ベース･オブジェクトのメッセージ･キュー９を検査する。

【０１１１】

メッセージ･キュー９にメッセージが格納されていれば、メッセージ･キューからメッセージを取り出し、スケジューラ６６にＩｎｖｏｋｅメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【０１１２】

他方、メッセージ･キュー９にメッセージが格納されていなければ、スケジューラ６６にＴｅｒｍｉｎａｔｅメッセージを送信して、ベース･オブジェクトの実行を終了させる。

【０１１３】

また、ｍＣＯＯＰＭａｉｌｅｒ６４は、メタ･オブジェクトから送られてくるＤｅｌｉｖｅｒメッセージにより起動して、所定の手続きを実行するようになっている。

【０１１４】

図６には、ｍＣＯＯＰＭａｉｌｅｒ６４がメタ･オブジェクトからＤｅｌｉｖｅｒメッセージを受信したことに応答して実行する手続きをフローチャート形式で示している。以下、このフローチャートに従って説明する。

【０１１５】

ｍＣＯＯＰＭａｉｌｅｒ６４は、Ｄｅｌｉｖｅｒメッセージで引数に指定された受信オブジェクトに対して、引数で指定されたメッセージを配送する。

【０１１６】

ステップＳ１１では、受信オブジェクトが別のメッセージに処理中であるか否かを検査する。

【０１１７】

別のメッセージを処理中でないときには、ステップＳ１２に進み、スケジューラ６６に対してＩｎｖｏｋｅメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【０１１８】

他方、受信側オブジェクトが別のメッセージを処理中である場合には、ステップＳ１４に進み、メッセージ･キュー９にメッセージを格納する。

【０１１９】

最後に、"Ｅｘｉｔ"ＡＰＩを発行して、その実行を終了させる（ステップＳ１３）。

【０１２０】

また、ｍＣＯＯＰＭａｉｌｅｒ６４は、メタ･オブジェクトから送られるＤｅｌｉｖｅｒＴａｇメッセージにより起動されると、引数で指定されたＴＩＤをＲＩＤに変換して、ＲＩＤに対応するＲＢｏｘを検索する。

【０１２１】

ＲＢｏｘに対して既に"Ｒｅｃｅｉｖｅ"ＡＰＩが発行されていれば、"Ｒｅｐｌｙ"ＡＰＩの引数で指定された結果メッセージを"Ｒｅｃｅｉｖｅ"ＡＰＩを発行したベース･オブジェクトが指定した領域に格納する。そして、スケジューラ６６にＲｅｓｕｍｅメッセージを送信して、"Ｒｅｃｅｉｖｅ"ＡＰＩを発行したベース･オブジェクトの実行を再開させる。

【０１２２】

他方、ＲＢｏｘに対して未だ"Ｒｅｃｅｉｖｅ"ＡＰＩが発行されていなければ、ＤｅｌｉｖｅｒＴａｇで指定された結果メッセージをＲＢｏｘに格納する。ｍＣＯＯＰＭａｉｌｅｒ６４は、"Ｅｘｉｔ"ＡＰＩによりその実行を終了させる。

【０１２３】

図４中に示されるｍＤｒｉｖｅＭａｉｌｅｒ６５は、ドライバ実行環境であるｍＤｒｉｖｅに対してメッセージ通信サービスを提供するオブジェクトであり、ｍＣＯＯＰＭａｉｌｅｒ６４と同様に、オペレーティング･システム５３を構成するオブジェクトの１つである。本実施例では、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７の構成オブジェクトとして実現されている。

【０１２４】

ｍＤｒｉｖｅＭａｉｌｅｒ６５は、以下に示すメッセージを受け取り、該当す

る各動作を実行するようになっている。

【０１２５】

【表２】

メッセージ	アクション
SendWithContinuation	1. Continuation を作成し、ContIDを返す。 2. 受信オブジェクトが同一実行環境であれば、受信オブジェクトにメッセージを配送する。 3. 受信オブジェクトが異なる実行環境であれば、その実行環境でメッセージを実現するメタ･オブジェクトに対しDeliverオブジェクトを送信する。
Kick	1. ContIDをTIDに変換する。 ２．ＴＩＤが同一実行環境のものであれば、Continuation に設定された継続オブジェクトに結果メッセージと継続メッセージを配送する。 3. TIDが異なる実行環境のものであれば、DeliverTagメッセージを送信する。
Exit	1. メッセージ･キューにメッセージがあれば、送信する。 2. そうでなければオブジェクトの実行を終了させる。
Deliver	メッセージを配送する。
DeliverTag	結果メッセージを配送する。

【０１２６】

ｍＤｒｉｖｅＭａｉｌｅｒ６５は、ｍＤｒｉｖｅのベース･オブジェクト（例えば、ドライバ５２）から送られるＳｅｎｄＷｉｔｈＣｏｎｔｉｎｕａｔｉｏｎメッセージによって起動されると、その内部の継続情報を格納するＣｏｎｔｉｎｕａｔｉｏｎを作成する。そして、Ｃｏｎｔｉｎａｕｔｉｏｎを識別するための識別子ＣｏｎｔＩＤを、メッセージ送信元であるベース･オブジェクトに返す。また、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、メッセージ受信側のオブジェクトが同一の実行環境上に存在するか否かを検査する。

【０１２７】

同一実行環境上であった場合には、さらに、該受信側オブジェクトが別のメッセージを処理中か否かを検査する。別のメッセージを処理中でない場合には、スケジューラ６６に対してＩｎｖｏｋｅメッセージを送信して、メッセージとＣｏ

ｎｔＩＤとともに受信オブジェクトを起動する。また、受信側のオブジェクトが別のメッセージを処理中であった場合には、メッセージ･キュー９にメッセージを格納する。

【０１２８】

他方、受信側のオブジェクトが異なる実行環境上に存在する場合には、その実行環境のメッセージ通信を実現しているメタ･オブジェクトに対して、Ｄｅｌｉｖｅｒメッセージを送信する。このとき、ＳｅｎｄＷｉｔｈＣｏｎｔｉｎｕａｔｉｏｎでベース･オブジェクトが指定したメッセージと、ＣｏｎｔＩＤをＴＩＤに変換したものも送信する。

【０１２９】

最後に、”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩによりベース･オブジェクトの実行を再開させる。

【０１３０】

また、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、ｍＤｒｉｖｅのベース･オブジェクトから送られるＫｉｃｋメッセージにより起動されると、”Ｋｉｃｋ”ＡＰＩの引数で指定されたＣｏｎｔＩＤをＴＩＤに変換して、ＴＩＤが同一実行環境上の識別子であるか否かを検査する。

【０１３１】

ＴＩＤが同一の実行環境上のものであった場合には、ＣｏｎｔＩＤよりＣｏｎｔｉｎｕａｔｉｏｎを検索する。そして、スケジューラ６６にＩｎｖｏｋｅメッセージを送信して、Ｃｏｎｔｉｎｕａｔｉｏｎに設定されている継続メッセージと結果メッセージとともに継続オブジェクトを起動する。

【０１３２】

他方、ＴＩＤが異なる実行環境のものであった場合には、ＴＩＤから結果メッセージを配送する実行環境のメッセージ通信を実現しているメタ･オブジェクトを特定して、そのメタ･オブジェクトに対してＤｅｌｉｖｅｒＴａｇメッセージを送信する。このとき、結果メッセージとＴＩＤを指定する。

【０１３３】

最後に、”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩによりベース･オブジェクトの実行を

再開させる。

【０１３４】

また、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、ｍＤｒｉｖｅのベース･オブジェクトから送られるＥｘｉｔメッセージにより起動されると、ベース･オブジェクトのメッセージ･キュー９を検査する。

【０１３５】

メッセージ･キュー９にメッセージが格納されていれば、メッセージ･キュー９からメッセージを取り出して、スケジューラ６６にＩｎｖｏｋｅメッセージを送信して、メッセージとともに受信側オブジェクトを起動する。また、メッセージ･キュー９にメッセージが格納されていなければ、スケジューラ６６にＴｅｒｍｉｎａｔｅメッセージを送信して、ベース･オブジェクトの実行を終了させる。

【０１３６】

また、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、メタ･オブジェクトから送られるＤｅｌｉｖｅｒメッセージにより起動されると、図６に示す手続きを実行するようになっている。

【０１３７】

ｍＤｒｉｖｅＭａｉｌｅｒ６５は、Ｄｅｌｉｖｅｒメッセージで引数に指定された受信オブジェクトに対して、引数で指定されたメッセージを配送する。

【０１３８】

ステップＳ１１では、受信オブジェクトが別のメッセージに処理中であるか否かを検査する。

【０１３９】

別のメッセージを処理中でないときには、ステップＳ１２に進み、スケジューラ６６に対してＩｎｖｏｋｅメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【０１４０】

他方、受信側オブジェクトが別のメッセージを処理中である場合には、ステップＳ１４に進み、メッセージ･キュー９にメッセージを格納する。

【０１４１】

最後に、"Ｅｘｉｔ"ＡＰＩを発行して、その実行を終了させる（ステップＳ１３）。

【０１４２】

また、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、メタ･オブジェクトから送られるＤｅｌｉｖｅｒＴａｇメッセージにより起動されると、引数で指定されたＴＩＤをＣｏｎｔＩＤに変換して、ＣｏｎｔＩＤに対応するＣｏｎｔｉｎｕａｔｉｏｎを検索する。そして、スケジューラ６６にＩｎｖｏｋｅメッセージを送信して、Ｃｏｎｔｉｎｕａｔｉｏｎに設定されている継続メッセージと結果メッセージとともに継続オブジェクトを起動する。

【０１４３】

図４中に示されるスケジューラ６６は、オブジェクトの実行スレッドのスケジューリングに関するサービスを提供するオブジェクトであり、ｍＣＯＯＰＭａｉｌｅｒ６４やｍＤｒｉｖｅＭａｉｌｅｒ６５と同様に、オペレーティング･システム５３を構成するオブジェクトの１つである。

【０１４４】

スケジューラ６６は、以下に示すメッセージを受け取り、該当する各動作を実行するようになっている。

【０１４５】

【表３】

メッセージ	アクション
Invoke	実行スレッドを起動する。
Resume	実行スレッドの実行を再開させる。
Terminate	実行スレッドの実行を終了させる。
ResumeBase	メタ･オブジェクトの実行を終了させ、ベース･オブジェクトの実行を再開させる。
ExitFromMetaCall	メタ･オブジェクトの実行を終了させ、ベース･オブジェクトの実行も停止状態のままにする。

【０１４６】

スケジューラ６６は、メタ･オブジェクトから送られるＩｎｖｏｋｅメッセージにより起動されると、引数で指定された実行スレッドを、所定のスケジューリ

ング･ポリシーに従ってスケジュールを行い起動する。

【０１４７】

また、スケジューラ６６は、メタ･オブジェクトから送られるＲｅｓｕｍｅメッセージにより起動されると、引数で指定された停止中の実行スレッドの実行を、所定のスケジューリング･ポリシーに従ってスケジュールを行い再開させる。

【０１４８】

また、スケジューラ６６は、メタ･オブジェクトから送られるＴｅｒｍｉｎａｔｅメッセージにより起動されると、引数で指定された実行スレッドの実行を終了させる。

【０１４９】

また、スケジューラ６６は、メタ･オブジェクトから送られるＲｅｓｕｍｅＢａｓｅメッセージにより起動されると、メタ･オブジェクトの実行を終了させ、メタ･コールを発行したベース･オブジェクトの実行を再開させる。

【０１５０】

また、スケジューラ６６は、メタ･オブジェクトから送られるＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌメッセージにより起動されると、メタ･オブジェクトの実行を終了させ、メタ･コールを発行したベース･オブジェクトも停止状態のままにする。

【０１５１】

図４中に示されるＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、オペレーティング･システム５３を構成する複合オブジェクトであり、上述したｍＣＯＯＰＭａｉｌｅｒ６４及びｍＤｒｉｖｅＭａｉｌｅｒ６５をその構成オブジェクトとする。

【０１５２】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、その内部の構成オブジェクトに対して、システム環境ｍＣｏｒｅと互換性のある”ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ”ＡＰＩを提供する。これによって、各構成オブジェクトｍＣＯＯＰＭａｉｌｅｒ６４及びｍＤｒｉｖｅＭａｉｌｅｒ６５を、通常のメタ･オブジェクトであるのと同様にプログラミングすることができる。

【０１５３】

また、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、"ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ"ＡＰＩを実現するために、構成オブジェクトのメッセージ通信履歴情報を保持するためのコール･ヒストリ８を内部に備えている。

【０１５４】

本実施例では、コール･ヒストリ８は、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７内の構成オブジェクトがＵｎｉｆｉｅｄＭａｉｌｅｒ５７外のオブジェクトからのメッセージ通信やメタ･コールにより起動されたときに、実行スレッドでの実行を示す"Ｅｘｔｅｒｎａｌ"に設定される。

【０１５５】

複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７がその構成オブジェクトに対して提供する"ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ"ＡＰＩについて、以下に説明する。

【０１５６】

Ｓｅｎｄ：

このＡＰＩは、送信側のメタ･オブジェクトから、引数で指定される受信側のメタ･オブジェクトに対して、引数で指定されるメッセージを送信して、引数で指定されるメソッドを起動する。

【０１５７】

図７には、"ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ"ＡＰＩであるＳｅｎｄの手続きをフローチャートの形式で示している。以下、このフローチャートに従い説明する。

【０１５８】

まず、ステップＳ２１では、受信側のオブジェクトが複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７内の構成オブジェクトであるか否かを検査する。

【０１５９】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７内の構成オブジェクトである場合には、ステップＳ２２に進み、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７内でメッセージ通信履歴情報を保持するコール･ヒストリ８を、同一実行スレッドでの実行を示すＩｎｔｅｒｎａｌに設定する。

【０１６０】

次いで、ステップＳ２３では、実行スレッドから受信オブジェクトのエントリを得る。そして、ステップＳ２４では、エントリにジャンプして実行を開始する。このとき、実行スレッドの切り替えを行わず、同一実行スレッドで処理を行う。このため、関数呼び出しと同等の処理が行われる。

【０１６１】

受信オブジェクトの実行から戻ってきたときには、コール･ヒストリの状態を１つ前の”Ｓｅｎｄ”ＡＰＩ呼び出し時の状態に戻して（ステップＳ２５）、”Ｓｅｎｄ”ＡＰＩ呼び出し後のポイントから送信オブジェクトの実行を再開する（ステップＳ２６）。

【０１６２】

また、受信オブジェクトがＵｎｉｆｉｅｄＭａｉｌｅｒ５７外のオブジェクトである場合には、ステップＳ２７において、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、その内部に保持するリクエスト･キュー９に、メッセージと受信オブジェクトに関する情報を格納する。

【０１６３】

Ｅｘｉｔ：

このＡＰＩは、”Ｓｅｎｄ”ＡＰＩで起動されたメタ･オブジェクトの実行を終了する。図８には、このＥｘｉｔの手続きをフローチャートの形式で示している。以下、このフローチャートに従い説明する。

【０１６４】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、まず、メッセージ通信履歴情報を保持するコール･ヒストリ８の状態を検査する（ステップＳ３１）。

【０１６５】

コール･ヒストリ８が同一実行スレッドでの実行を示すＩｎｔｅｒｎａｌに設定されている場合には、関数の復帰処理と同等の処理を行い（ステップＳ３２）、呼び出し元の構成オブジェクトに制御を移す。

【０１６６】

また、コール･ヒストリ８が異なる実行スレッドでの実行を示すＥｘｔｅｒｎ

ａｌで設定されている場合には、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７が保持するリクエスト･キュー９に格納されているメッセージを受信オブジェクト毎に１つにまとめて（ステップＳ３３）、ｍＣｏｒｅの”Ｓｅｎｄ”ＡＰＩを用いてメッセージの送信を行う。そして、ｍＣｏｒｅの”Ｅｘｉｔ”ＡＰＩを発行して、メタ･オブジェクトの実行を終了させる（ステップＳ３４）。

【０１６７】

ＲｅｓｕｍｅＢａｓｅ：

このＡＰＩは、メタ･コールにより起動されたメタ･オブジェクトの実行を終了し、メタ･コールを発行したベース･オブジェクトの実行を再開する。図９には、このＲｅｓｕｍｅＢａｓｅの手続きをフローチャートの形式で示している。以下、このフローチャートに従い説明する。

【０１６８】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、まず、メッセージ通信履歴情報を保持するコール･ヒストリ８の状態を検査する（ステップＳ４１）。

【０１６９】

コール･ヒストリ８が同一実行スレッドでの実行を示すＩｎｔｅｒｎａｌに設定されている場合、ステップＳ４２に進み、スケジューラ６６を受信オブジェクトに指定して、ＲｅｓｕｍｅＢａｓｅメッセージをＵｎｉｆｉｅｄＭａｉｌｅｒ５７が保持するリクエスト･キュー９に格納する。そして、リクエスト･キュー９に格納されているメッセージを受信オブジェクト毎に１つにまとめて、ｍＣｏｒｅの”Ｓｅｎｄ”ＡＰＩを用いてメッセージの送信を行う（ステップＳ４３）。スケジューラ６６へＲｅｓｕｍｅＢａｓｅメッセージが送信されるため、この実行によりメタ･コール処理が終了し、メタ･オブジェクトの実行は終了して、同時にベース･オブジェクトの実行が再開する。

【０１７０】

他方、コール･ヒストリ８が異なる実行スレッドでの実行を示すＥｘｔｅｒｎａｌに設定されている場合、ステップＳ４４に進んで、エラーを発生させる。

【０１７１】

ＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌ：

このＡＰＩは、図９に示したものと同様の手順に従い、メタ･コールにより起動されたメタ･オブジェクトの実行を終了する。但し、メタ･コールを発行したベース･オブジェクトの実行は停止したままにする。

【０１７２】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、まず、メッセージ通信履歴情報を保持するコール･ヒストリ８の状態を検査する（ステップＳ４１）。

【０１７３】

コール･ヒストリ８が同一実行スレッドでの実行を示すＩｎｔｅｒｎａｌに設定されている場合、ステップＳ４２では、ＲｅｓｕｍｅＢａｓｅメッセージとなっているが、その代わりにＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌメッセージを、スケジューラ６６を受信オブジェクトに指定してＵｎｉｆｉｅｄＭａｉｌｅｒ５７が保持するリクエスト･キュー９に格納する。そして、ステップＳ４３では、リクエスト･キュー９に格納されているメッセージを受信オブジェクト毎に１つにまとめて、ｍＣＯｒｅの”Ｓｅｎｄ”ＡＰＩを用いてメッセージの送信を行う。スケジューラ６６へはＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌメッセージが送信されるため、この実行によりメタ･コール処理が終了し、メタ･オブジェクトの実行は終了して、同時にベース･オブジェクトの実行もメタ･コール発行時点で停止したままになる。

【０１７４】

他方、コール･ヒストリ８が異なる実行スレッドでの実行を示すＥｘｔｅｒｎａｌに設定されている場合、ステップＳ４４に進んで、エラーを発生させる。

【０１７５】

次いで、本実施例に係るオブジェクト指向オペレーティング･システムが提供する、異なる実行環境間でメッセージ通信を行う場合のシナリオについて説明する。

【０１７６】

図１０には、実行環境ｍＣＯＯＰ上で動作するアプリケーションから他の実行環境ｍＤｒｉｖｅ上で動作するドライバに対して、ｍＣＯＯＰにより提供される”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩを用いてメッセージ通信を行う場合の各オブ

ジェクト間の処理手順を示している。但し、図面の簡素化のため、メッセージ返信の処理については省略している。

【０１７】

アプリケーションは、ｍＣＯＯＰで提供される”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩを処理Ｐ１で発行する。このとき、”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩはメタ･コールを発行するため、ベース･オブジェクトの実行スレッドからメタ･オブジェクトの実行スレッドへの遷移が行われる。また、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７外からのメタ･コールであるため、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７におけるメッセージ通信履歴情報を保持するコール･ヒストリ８は、外部の実行スレッドからの実行であることを示すＥｘｔｅｒｎａｌに設定される。

【０１７８】

ｍＣＯＯＰＭａｉｌｅｒ６４がＳｅｎｄＷｉｔｈＲＢｏｘメッセージを受け取ると、次の処理Ｐ２において、図５に示す手続きを行う。ｍＣＯＯＰＭａｉｌｅｒ６４は、まず、ステップＳ１では、結果メッセージを格納するためのＲＢｏｘを作成する。そして、ＲＢｏｘを識別するためのＲＩＤ識別子をメッセージ送信オブジェクトのＲＩＤ格納領域に設定し、”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩの終了時にアプリケーションがＲＩＤを得られるようにする。ＲＩＤ格納領域は、”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩの引数として、アプリケーションにより与えられる。

【０１７９】

次いで、ステップＳ２では、メッセージ受信側のオブジェクトが同一の実行環境上に存在するか否かを検査する。これは、ＯＩＤを用いてオブジェクト･テーブル７０から得られるオブジェクト記述情報を検索することで行われる。

【０１８０】

図１０に示すシナリオでは、受信側オブジェクトであるドライバは実行環境ｍＤｒｉｖｅ上のオブジェクトであるので、ステップＳ２では、送信側の実行環境ｍＣＯＯＰとは異なる実行環境上のオブジェクトとして判断される。このため、判断ブロックの分岐Ｎｏからステップ７に進む。

【０１８１】

ステップＳ７では、ｍＣＯＯＰＭａｉｌｅｒは、受信側のオブジェクトの実行環境上でメッセージ通信を実現するメタ･オブジェクトに対して、ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅのＡＰＩである”Ｓｅｎｄ”を用いてＤｅｌｉｖｅｒメッセージを送信する。

【０１８２】

図１０に示すシナリオでは、ｍＤｒｉｖｅＭａｉｌｅｒ６５に対してＳｅｎｄを発行することになる。Ｄｅｌｉｖｅｒメッセージには、”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩの引数で指定されたメッセージとＲＩＤをＴＩＤに変換したものを付加する。

【０１８３】

複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７が提供するＵｎｉｆｉｅｄＭａｉｌｅｒＣｏｒｅの”Ｓｅｎｄ”ＡＰＩは、図７に示す手続きを行う。

【０１８４】

まず、ステップＳ２１では、受信側のオブジェクトが複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７内の構成オブジェクトであるか否かを検査する。この検査は、ＯＩＤを用いてオブジェクト･テーブル７０を検索して得られるオブジェクト記述情報を使うことで行われる。

【０１８５】

図１０に示すシナリオでは、受信側のオブジェクトｍＤｒｉｖｅＭａｉｌｅｒ６５は、同一の複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７内の構成オブジェクトであるので、次ステップＳ２２に進む。

【０１８６】

ステップＳ２２では、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７内でメッセージ通信履歴情報を保持するコール･ヒストリ８を、同一実行スレッドでの実行を示すＩｎｔｅｒｎａｌに設定する。

【０１８７】

次いで、ステップＳ２３では、実行スレッドから受信オブジェクトのエントリを得る。そして、ステップＳ２４では、エントリにジャンプして実行を開始する

。図１０中において、遷移Ｍ２として示される点線矢印がこの遷移に該当する。ここで、遷移Ｍ２を点線で示しているのは、実際には実行スレッドの切り替えが行われていないことを意味している。

【０１８８】

ｍＤｒｉｖｅＭａｉｌｅｒ６５がＤｅｌｉｖｅｒメッセージを受け取ると、図１０の処理Ｐ３では図６に示す手続きが行われる。

【０１８９】

ステップＳ１１では、受信オブジェクトが別のメッセージに処理中であるか否かを検査する。この検査は、受信オブジェクトようのメッセージ･キュー９が空であるか否かで判断することができる。そして、現在メッセージの処理を行っていなければ、ｍＤｒｉｖｅＭａｉｌｅｒ６５は、ステップＳ１２に進んで、ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの”Ｓｅｎｄ”ＡＰＩを用いて、スケジューラ６６に対してＩｎｖｏｋｅメッセージを送信する。

【０１９０】

ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅが提供する”Ｓｅｎｄ”ＡＰＩは、図７に示す手続きを行う。

【０１９１】

まず、ステップＳ２１では、受信側のオブジェクトが複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７内の構成オブジェクトであるか否かを検査する。

【０１９２】

図１０に示すシナリオでは、受信オブジェクトとしてのスケジューラ６６は、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ５７外の通常のオブジェクトであるので、ステップＳ２７に進む。

【０１９３】

ステップＳ２７では、受信オブジェクトを示すＯＩＤとメッセージがＵｎｉｆｉｅｄＭａｉｌｅｒ５７内のリクエスト･キュー９に格納される。

【０１９４】

図１０に示す遷移Ｍ３及びＭ４はいずれも点線矢印で示されているが、これらは、実際には実行スレッドの切り替えが行われていないことを意味する。すなわ

ち、この時点では、複合オブジェクト５７内の構成オブジェクトの観点からはスケジューラ６６へのメッセージ送信が実行されたことになるが、実際には、メッセージ通信は透過的に遅延される。したがって、図１０の処理Ｐ４は、この時点では未だ処理が行われない。

【０１９５】

ｍＤｒｉｖｅＭａｉｌｅｒ６５は、図１０中の処理Ｐ３でのＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの"Ｓｅｎｄ"ＡＰＩから戻ると、処理Ｐ５において、Ｄｅｌｉｖｅｒメッセージの処理の続きを行う。すなわち、図６のステップＳ１３において、ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの"Ｅｘｉｔ"ＡＰＩを発行する。

【０１９６】

ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅが提供する"Ｅｘｉｔ"ＡＰＩは、図８に示す手続きを行う。

【０１９７】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、まず、メッセージ通信履歴情報を保持するコール･ヒストリ８の状態を検査する（ステップＳ３１）。

【０１９８】

図１０に示すシナリオでは、ｍＣＯＯＰＭａｉｌｅｒ６４からｍＤｒｉｖｅＭａｉｌｅｒ６５に対してＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの"Ｓｅｎｄ"ＡＰＩを発行したときに、図７のステップＳ２２においてＩｎｔｅｒｎａｌが設定されている。したがって、ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの"Ｅｘｉｔ"ＡＰＩは、図８のステップＳ３２に進む。

【０１９９】

ステップＳ３２では、ＵｎｉｆｉｅｄＭａｉｌｍＣｏｒｅの"Ｓｅｎｄ"ＡＰＩの発行元に実行を復帰させる。これは、関数の復帰と同様の処理で行われる。復帰処理により、ｍＣＯＯＰＭａｉｌｅｒ６４は、図１０の処理Ｐ２におけるＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの"Ｓｅｎｄ"ＡＰＩの呼び出し手続きの続きを行う。

【０２００】

図１０に示す遷移Ｍ５は点線矢印で示されているが、これらは、実際には実行スレッドの切り替えが行われていないことを意味する。

【０２０１】

この結果、図７のステップＳ２５に実行が移る。ステップＳ２５では、コール・ヒストリの状態を１つ前の状態に戻す。図１０に示すシナリオの場合、コール・ヒストリ８の状態は、ｍＣＯＯＰＭａｉｌｅｒ６４からｍＤｒｉｖｅＭａｉｌｅｒ６５に対してＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの”Ｓｅｎｄ”ＡＰＩを発行したときに設定されたＩｎｔｅｒｎａｌ状態（図７のステップＳ２２を参照のこと）から、アプリケーションがメタ・コールとしてｍＣＯＯＰの”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩを発行してＵｎｉｆｉｅｄＭａｉｌｅｒ５７に実行スレッドが切り替わったときに設定されたＥｘｔｅｒｎａｌ状態に戻る。すなわち、図５のステップＳ５において、ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩを発行する。

【０２０２】

ｍＣＯＯＰＭａｉｌｅｒ６４が提供するＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩは、図９に示す手続きを行う。

【０２０３】

ＵｎｉｆｉｅｄＭａｉｌｅｒ５７は、まず、メッセージ通信履歴情報を保持するコール・ヒストリ８の状態を検査する（ステップＳ４１）。

【０２０４】

図１０に示すシナリオでは、アプリケーションがｍＣＯＯＰの”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩを発行した時点の状態、すなわち、メタ・コールとしてＵｎｉｆｉｅｄＭａｉｌｅｒ５７に実行スレッドが切り替わったときに設定されたＥｘｔｅｒｎａｌに状態が戻されているため、ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅの”ＲｅｓｕｍｅＢａｓｅ”ＡＰＩはステップＳ４２に進む。

【０２０５】

ステップＳ４２では、スケジューラ６６のＯＩＤとスケジューラ６６へのＲｅｓｕｍｅＢａｓｅメッセージをＵｎｉｆｉｅｄＭａｉｌｅｒ５７内のリクエスト・キュー９に格納する。そして、ステップＳ４３では、ＵｎｉｆｉｅｄＭａｉｌ

ｅｒ５７内のリクエスト･キュー９に格納されているメッセージを受信オブジェクト毎に１つにまとめて、ｍＣｏｒｅの”Ｓｅｎｄ”ＡＰＩを用いてメッセージの送信を行う。

【０２０６】

図１０に示すシナリオでは、遷移Ｍ３で送信されず遅延されていたスケジューラ６６へのＩｎｖｏｋｅメッセージと、図９のステップＳ４２で格納したスケジューラ６６へのＲｅｓｕｍｅＢａｓｅメッセージが１つにまとめられて、１回のｍＣｏｒｅの”Ｓｅｎｄ”ＡＰＩを用いてスケジューラ６６に送られることになる。

【０２０７】

スケジューラ６６は、ＩｎｖｏｋｅとＲｅｓｕｍｅＢａｓｅの２つのメッセージを受け取ると、図１０の処理Ｐ７において、本来は処理Ｐ４において行われるはずが遅延されていたＩｎｖｏｋｅと、新たなＲｅｓｕｍｅＢａｓｅの処理を行う。

【０２０８】

スケジューラ６６は、Ｉｎｖｏｋｅ処理を受けてスケジューリングを行い、遷移Ｍ７で通信先オブジェクトとなるドライバの実行スレッドを起動する。これによって、実行環境ｍＤｒｉｖｅ上で動作するドライバは、異なる実行環境ｍＣＯＯＰ上で動作するアプリケーションからｍＣＯＯＰのＡＰＩ”ＳｅｎｄＷｉｔｈＲＢｏｘ”で送られたメッセージを受け取り、該メッセージを処理Ｐ８で処理を行うことができる。

【０２０９】

また、スケジューラ６６は、ＲｅｓｕｍｅＢａｓｅの処理を行い、ＵｎｉｆｉｅｄＭａｉｌｅｒ５７内のメタ･オブジェクトの実行を終了させるとともに、図１０には示していないが、スケジューリング結果の適切な時点でアプリケーションの実行を再開させる。この結果、アプリケーションは、ｍＣＯＯＰが提供するＡＰＩ”ＳｅｎｄＷｉｔｈＲＢｏｘ”のメタ･コール呼び出しから復帰して、以後の処理を継続することができる。

【０２１０】

以上を総括すれば、メッセージ通信が頻繁に発生するような複数のオブジェクトを複合オブジェクトとして包括するデータ通信方式に対して、さらに、複合オブジェクト外へのメッセージ通信を一時的に格納するリクエスト･キュー９と構成オブジェクト間のメッセージ通信履歴情報を保持するコール･ヒストリ８を追加する。そして、構成オブジェクトｍＣＯＯＰＭａｉｌｅｒ及びｍＤｒｉｖｅＭａｉｌｅｒが複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒ外のオブジェクトであるスケジューラ６６に対してメッセージ通信を行うと、該メッセージを、受信オブジェクトとしてのスケジューラを示すＯＩＤとともにリクエスト･キュー９に一旦格納してッセージ通信を遅延させる。その後、構成オブジェクトの実行終了時にコール･ヒストリの状態の検査を行い、それが外部の実行スレッドからの実行を示すＥｘｔｅｒｎａｌであった場合には、リクエスト･キュー９に格納された複数のメッセージを各受信オブジェクト毎に１つにまとめて、１回のメッセージ通信で送信を行うことができる。つまり、各構成オブジェクトｍＣＯＯＰＭａｉｌｅｒ及びｍＤｒｉｖｅＭａｉｌｅｒ間でメッセージ通信を行っている期間中は、外部のオブジェクトへのメッセージ通信を遅延させる。そして、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒとしての動作が終了したときに、複数のメッセージをまとめて送信することで、メタ･オブジェクト間のメッセージ通信回数を削減し、ｍＣＯＯＰとｍＤｒｉｖｅという異なる実行環境間でのメッセージ通信サービスの性能を向上させることができるという訳である。

【０２１１】

本発明によれば、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒの各構成オブジェクトｍＣＯＯＰＭａｉｌｅｒ及びｍＤｒｉｖｅＭａｉｌｅｒ間でメッセージ通信を行う際に、複合オブジェクト外のオブジェクトであるスケジューラに対するメッセージが発生するにも拘わらず、コストを削減することができる。図１０に示す例では、遷移Ｍ２及びＭ５の間に挿入された遷移Ｍ３及びＭ４を、複合オブジェクト内の構成オブジェクトからは透過的に削除することができる。通常のオブジェクトのプログラミングを全く変更せずに、複合オブジェクトＵｎｉｆｉｅｄＭａｉｌｅｒに構成オブジェクトとしてリクエスト･キューやコール･ヒストリを追加するだけで実装することができる。

【０２１２】

すなわち、本発明によれば、メッセージ通信が頻繁に発生するような複数のオブジェクトを複合オブジェクトとして包括するデータ通信方式によって、メッセージ通信のコストを削減することができる訳である。特に、複合オブジェクト内の異なる構成オブジェクトで発行されたメッセージ通信であっても、それを遅延させて１つのメッセージ通信として送信することができる。

【０２１３】

［追補］

以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。

【０２１４】

【発明の効果】

以上詳記したように、本発明によれば、複数のオブジェクト間で好適にメッセージ交換を行うことができる、優れたデータ通信方式を提供することができる。

【０２１５】

また、本発明によれば、複数の並列オブジェクトで構成されるオブジェクト指向オペレーティング・システムにおいてオブジェクト間で好適にメッセージ交換を行うことができる、優れたデータ通信方式を提供することができる。

【０２１６】

また、本発明によれば、コンテキスト・スイッチなどのコスト発生を可能な限り抑制して、オブジェクト間のメッセージ交換を高速化することができる、優れたデータ通信方式を提供することができる。

【０２１７】

また、本発明によれば、オブジェクト指向オペレーティング・システムにおけるシステム構成の柔軟性や動的変更の容易性などの優位性を維持したままで、オブジェクト間のメッセージ通信時のコストを削減して、システム全体の実行性能

を向上させることができる。

【０２１８】

また、本発明によれば、複合オブジェクト内のオブジェクトと複合オブジェクト外に存在するオブジェクトとのメッセージ通信が頻繁に行われている場合、複合オブジェクト内の各オブジェクトには透過的な形式でメッセージが蓄積され、その後の適切なタイミングで蓄積された複数のメッセージをまとめて１回のメッセージ通信による送信することができる。この結果、メッセージ通信のコストを大幅に削減することができる。また、オブジェクトの独立性を保持したままシステム全体の実行性能を向上させることができる。

【図面の簡単な説明】

【図１】

本発明の実施に供されるデータ処理システム１０のハードウェア構成を模式的に示した図である。

【図２】

本実施例に係るデータ処理システム１０が有するソフトウェア環境を模式的に示した図である。

【図３】

オブジェクト指向オペレーティング・システムが提供する異なる実行環境間で行われるメッセージ通信の動作を示した図であり、より具体的には、実行環境ｍＣＯＯＰ上で動作するアプリケーションと実行環境ｍＤｒｉｖｅ上で動作するデバイス・ドライバとの間でメッセージ通信が行われる様子を示した図である。

【図４】

実行環境ｍＣＯＯＰ上で動作するアプリケーションと実行環境ｍＤｒｉｖｅ上で動作するデバイス・ドライバとの間でメッセージ通信が行われる様子を示した図であり、より具体的には、メッセージ通信時のオペレーティング・システム内での動作を詳細に示した図である。

【図５】

メタ・オブジェクトｍＣＯＯＰＭａｉｌｅｒ６４がベース・オブジェクトからＳｅｎｄＷｉｔｈＲＢｏｘメッセージを受信したことに応答して実行する手続きを

示したフローチャートである。

【図６】

メタ･オブジェクトｍＣＯＯＰＭａｉｌｅｒ６４又はｍＤｒｉｖｅＭａｉｌｅｒ６５がベース･オブジェクトからＤｅｌｉｖｅｒメッセージを受信したことに応答して実行する手続きを示したフローチャートである。

【図７】

”ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ”ＡＰＩであるＳｅｎｄの手続きを示したフローチャートである。

【図８】

”ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ”ＡＰＩであるＥｘｉｔの手続きを示したフローチャートである。

【図９】

”ＵｎｉｆｉｅｄＭａｉｌｅｒｍＣｏｒｅ”ＡＰＩであるＲｅｓｕｍｅＢａｓｅ及びＥｘｉｔＦｒｏｍＭｅｔａＣａｌｌの手続きを示したフローチャートである。

【図１０】

実行環境ｍＣＯＯＰ上で動作するアプリケーションから他の実行環境ｍＤｒｉｖｅ上で動作するドライバに対して、ｍＣＯＯＰに提供される”ＳｅｎｄＷｉｔｈＲＢｏｘ”ＡＰＩを用いてメッセージ通信を行う場合の各オブジェクト間の処理手順を示した図である。

【符号の説明】

１…複合オブジェクト

２，３…構成オブジェクト

４…構成オブジェクト実行環境

５…オブジェクト･テーブル

６…オブジェクト

７…実行環境

８…コール･ヒストリ

９…リクエスト･キュー

１０…データ処理システム

１１…プロセッサ

１２…メモリ

１３…ディスプレイ・コントローラ

１４…入力機器インターフェース

１５…ネットワーク・インターフェース

１６…ＨＤＤインターフェース

２１…ディスプレイ

２２…キーボード

２３…マウス

２４…ＨＤＤ

５１…アプリケーション

５２…ドライバ

５３…オペレーティング・システム

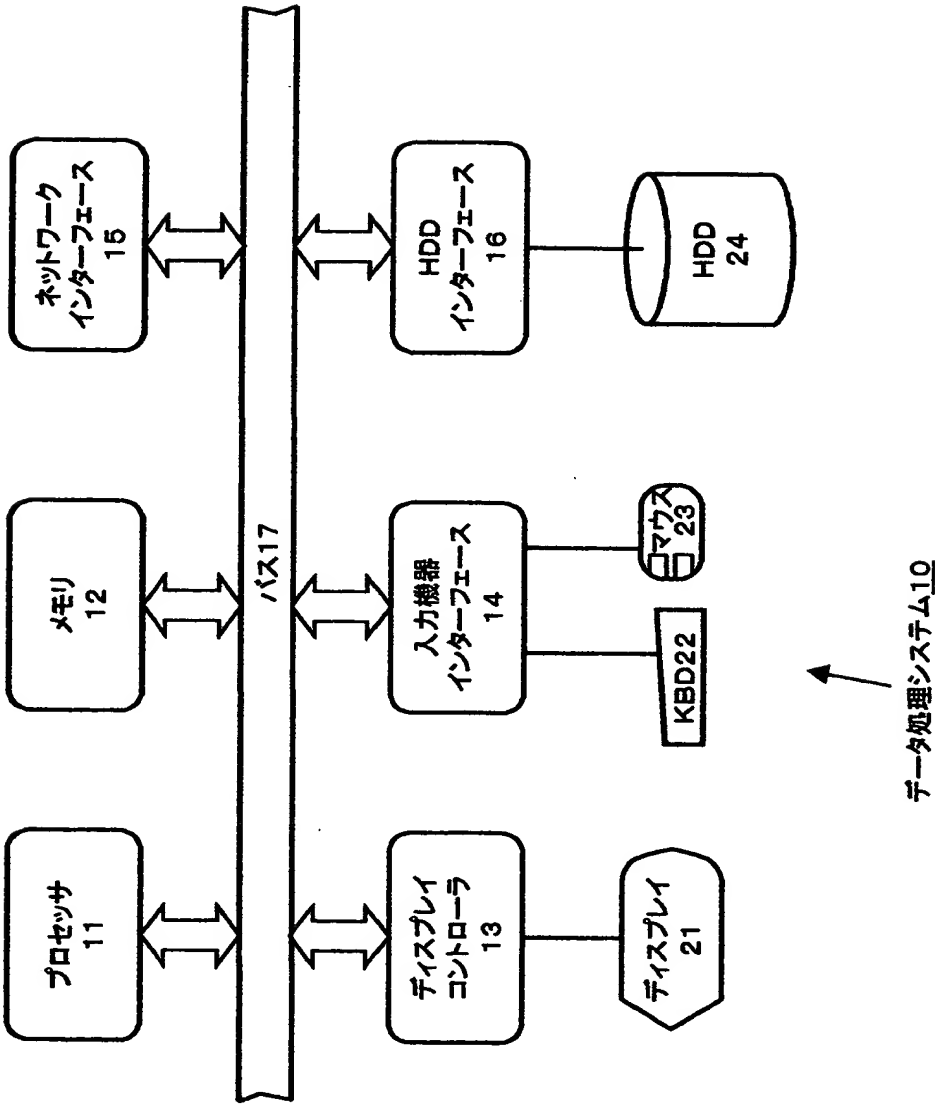
６４…ｍＣＯＯＰＭａｉｌｅｒ

６５…ｍＤｒｉｖｅＭａｉｌｅｒ

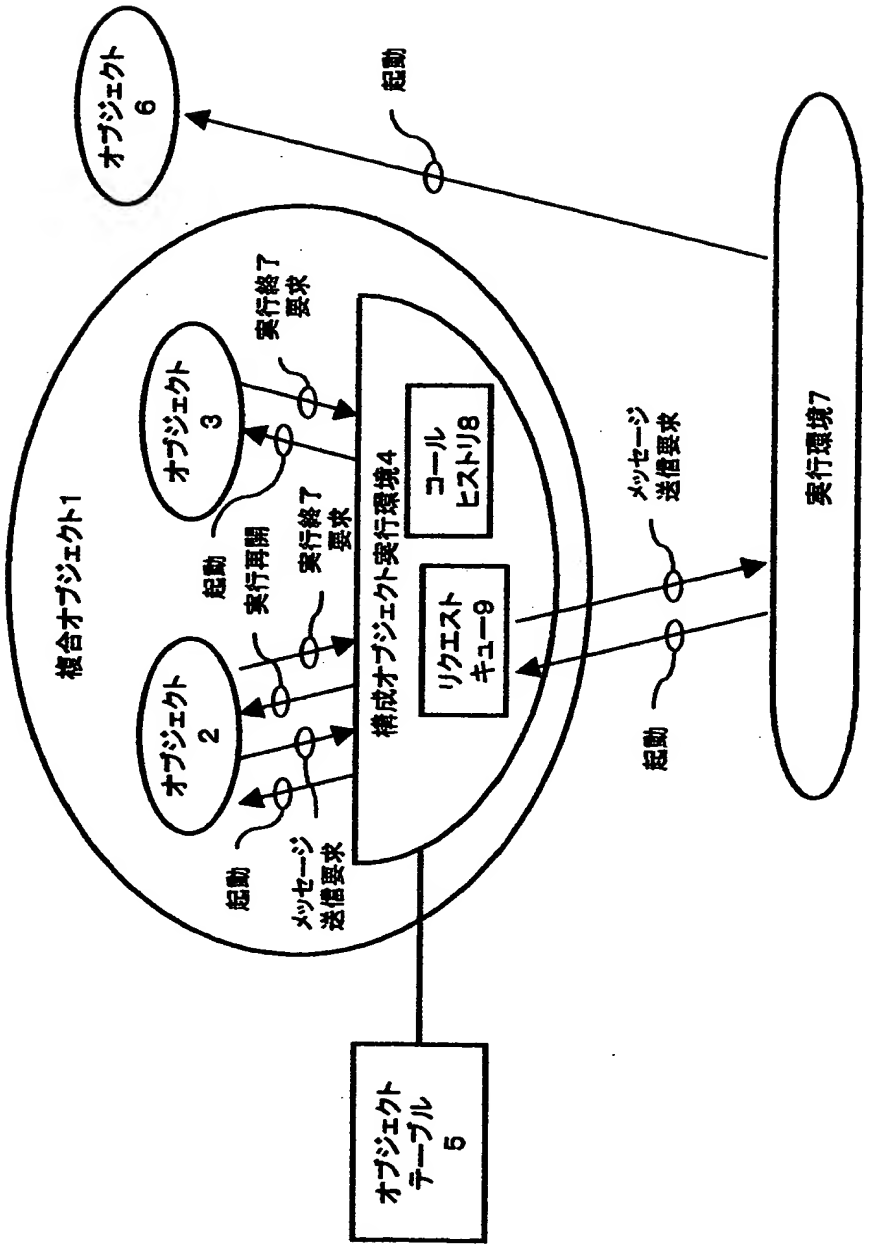
６６…スケジューラ

【書類名】　　図面

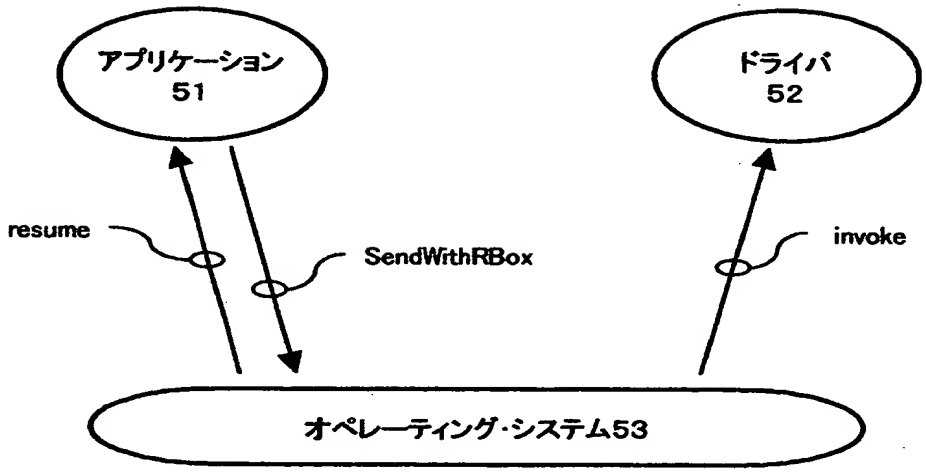
【図１】


ネットワーク
インターフェース
15
HDD
インターフェース
16
HDD
24
バス17
メモリ
12
入力機器
インターフェース
14
マウス
23
KBD22
プロセッサ
11
ディスプレイ
コントローラ
13
ディスプレイ
21
データ処理システム10

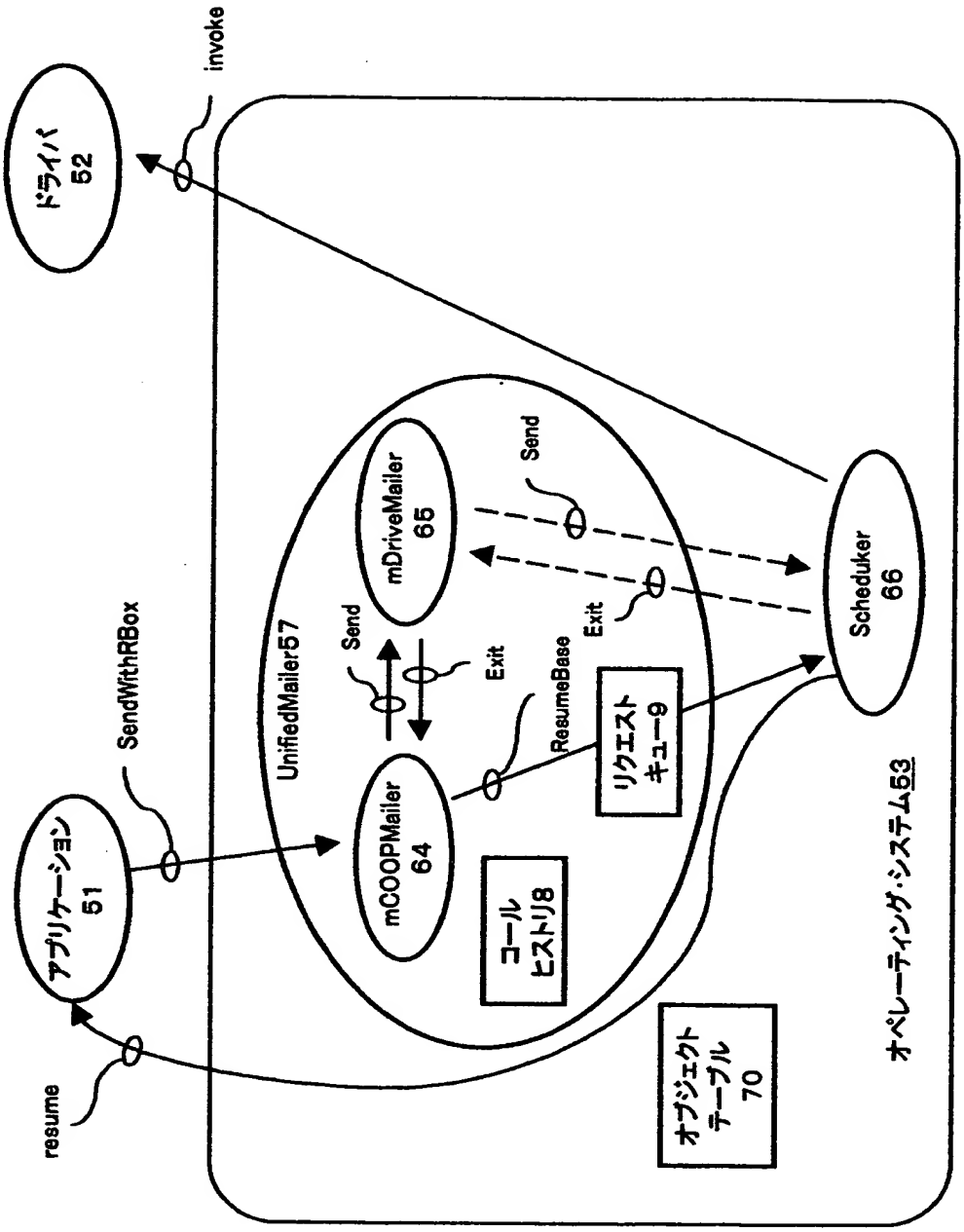
【図２】


オブジェクト
6
起動
実行環境7
メッセージ
送信要求
起動
複合オブジェクト1
オブジェクト
3
実行終了
要求
起動
実行再開
実行終了
要求
構成オブジェクト実行環境4
コール
ヒストリ8
リクエスト
キュー9
オブジェクト
2
起動
メッセージ
送信要求
オブジェクト
テーブル
5

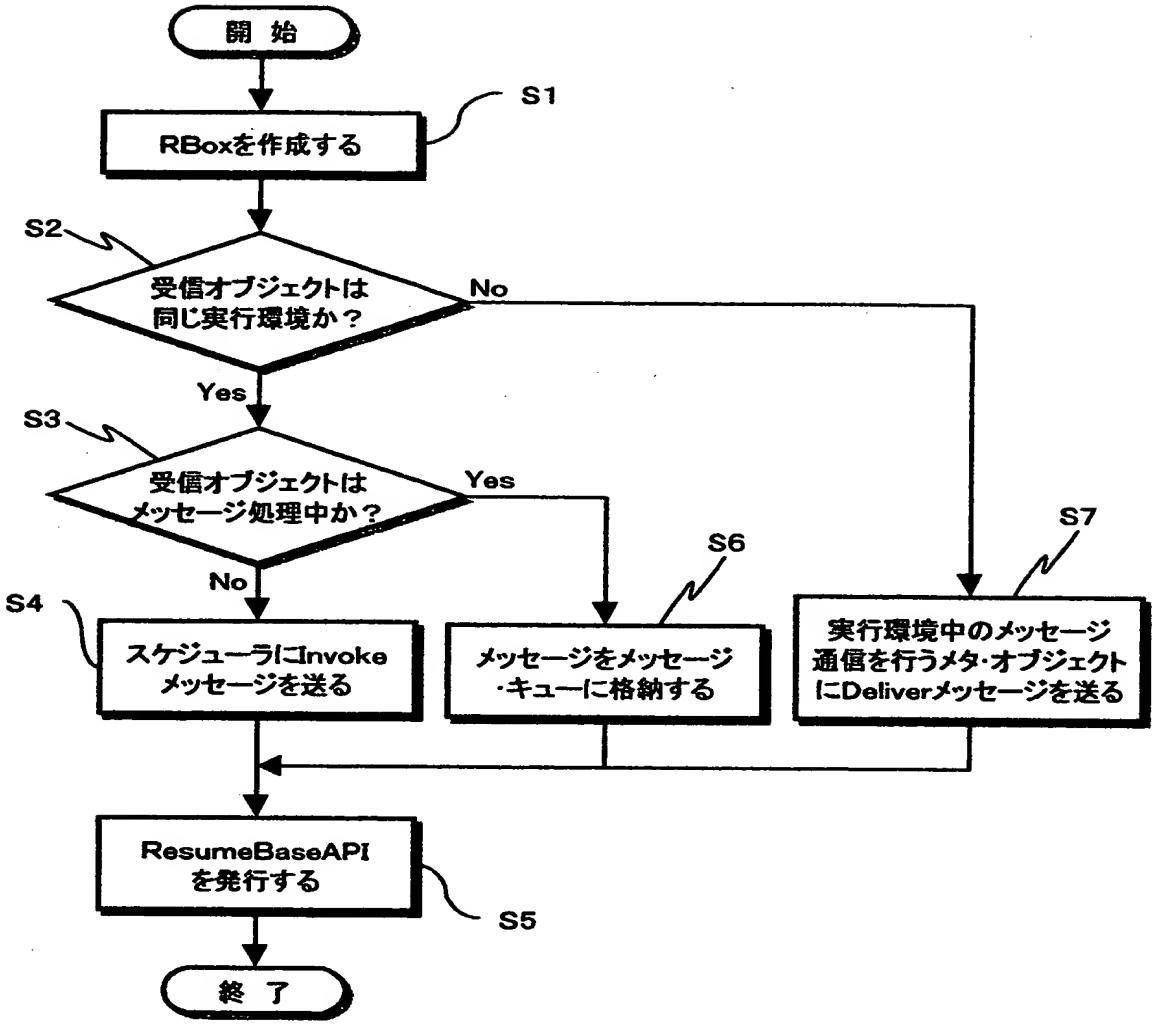
【図３】


アプリケーション
51
ドライバ
52
resume
SendWithRBox
invoke
オペレーティング・システム53

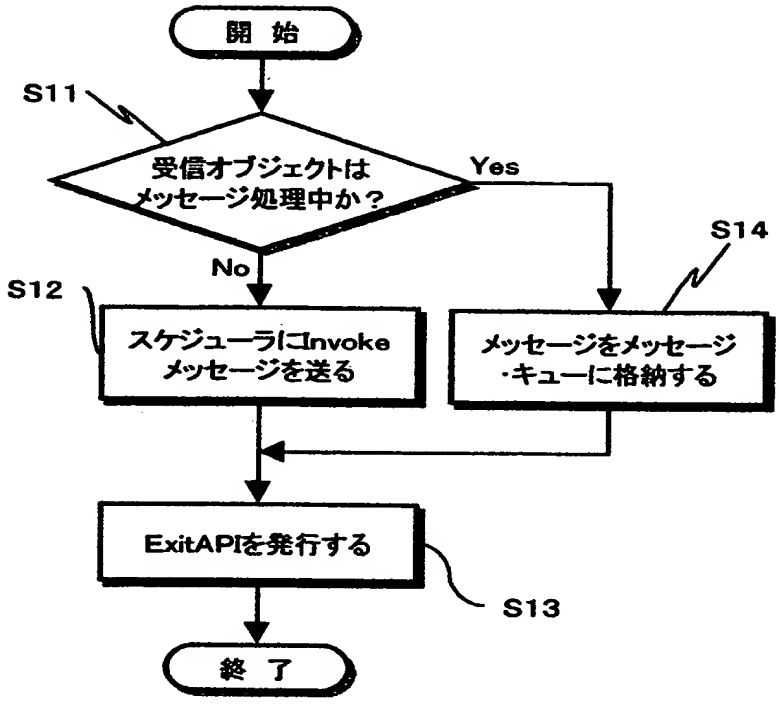
【図４】


ドライバ
52
invoke
アプリケーション
51
SendWithRBox
resume
UnifiedMailer57
mCOOPMailer
64
mDriveMailer
65
Send
Exit
Send
Exit
ResumeBase
リクエスト
キュー9
コール
ヒストリ8
Scheduker
66
オブジェクト
テーブル
70
オペレーティング・システム53

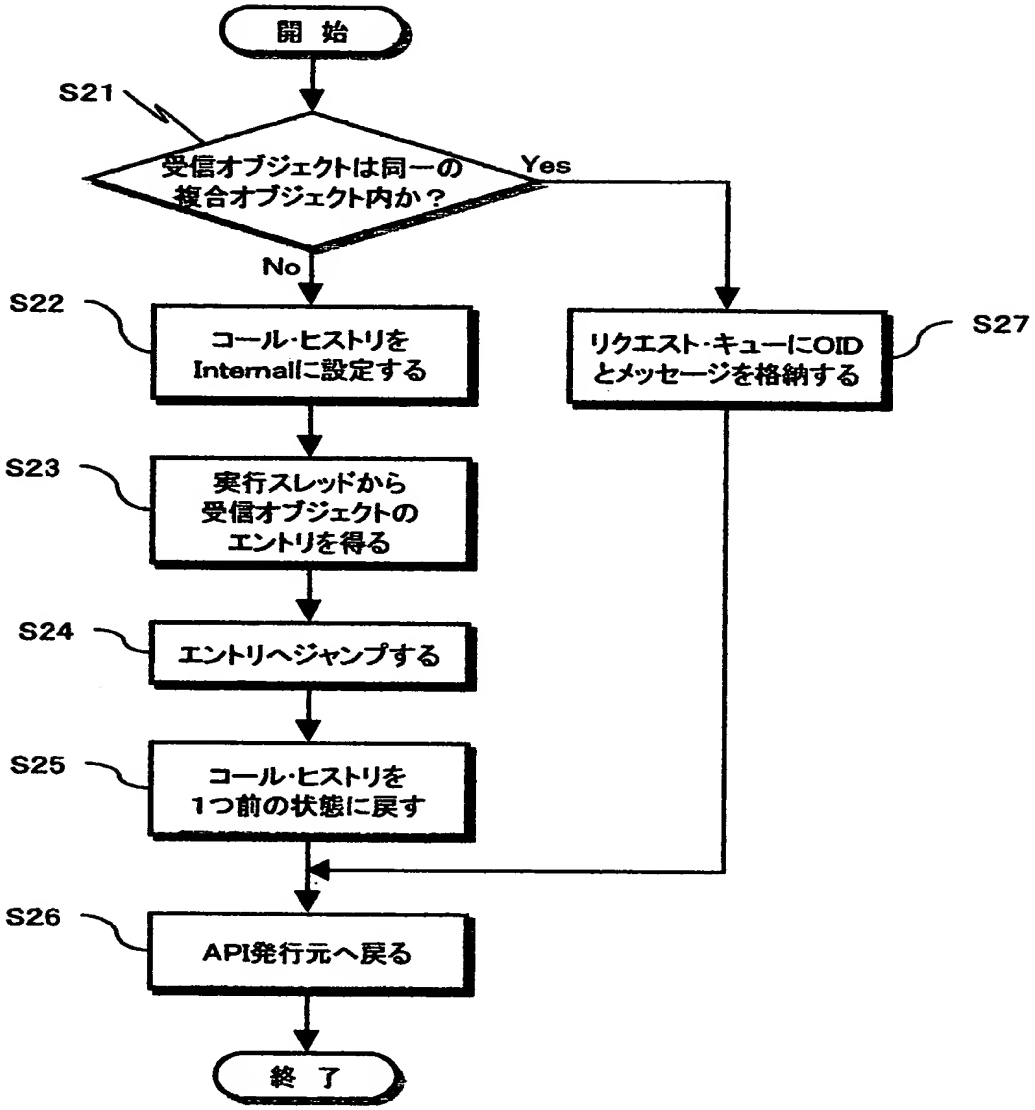
【図５】


開 始
S1
RBoxを作成する
S2
受信オブジェクトは
同じ実行環境か？
No
Yes
S3
受信オブジェクトは
メッセージ処理中か？
Yes
No
S4
スケジューラにInvoke
メッセージを送る
S6
メッセージをメッセージ
・キューに格納する
S7
実行環境中のメッセージ
通信を行うメタ・オブジェクト
にDeliverメッセージを送る
ResumeBaseAPI
を発行する
S5
終 了

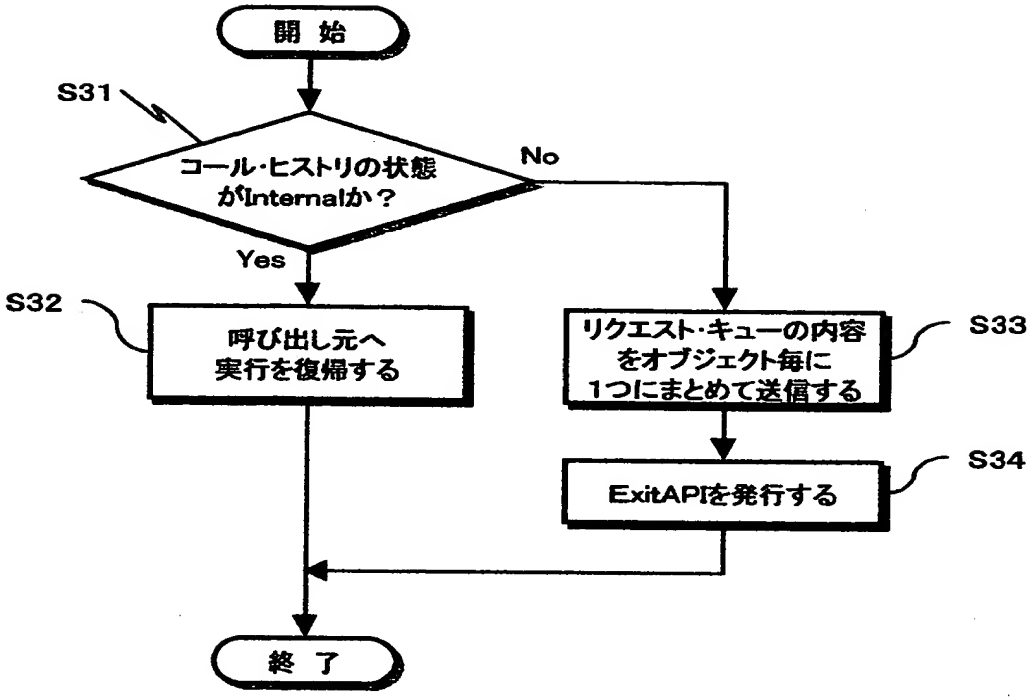
【図６】


開 始
S11
受信オブジェクトは
メッセージ処理中か？
Yes
No
S14
S12
スケジューラにInvoke
メッセージを送る
メッセージをメッセージ
・キューに格納する
ExitAPIを発行する
S13
終 了

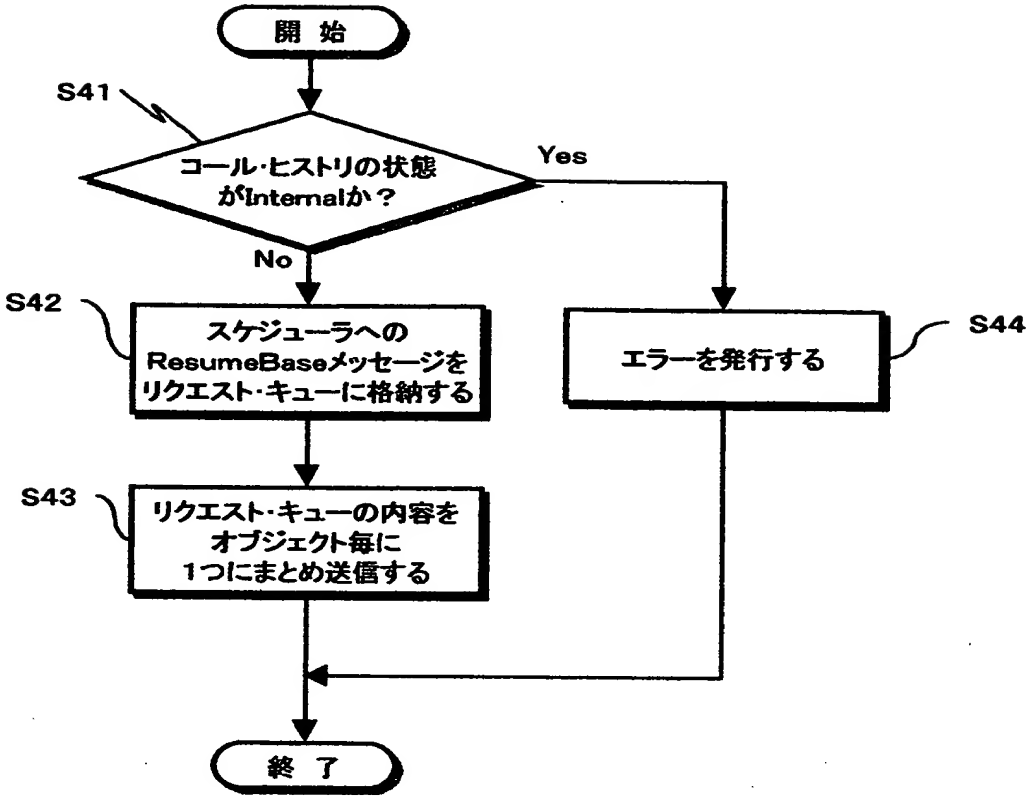
【図７】


開 始
S21
受信オブジェクトは同一の
複合オブジェクト内か？
Yes
No
S22
コール･ヒストリを
Internalに設定する
S23
実行スレッドから
受信オブジェクトの
エントリを得る
S24
エントリへジャンプする
S25
コール･ヒストリを
1つ前の状態に戻す
S26
API発行元へ戻る
終 了
S27
リクエスト･キューにOID
とメッセージを格納する

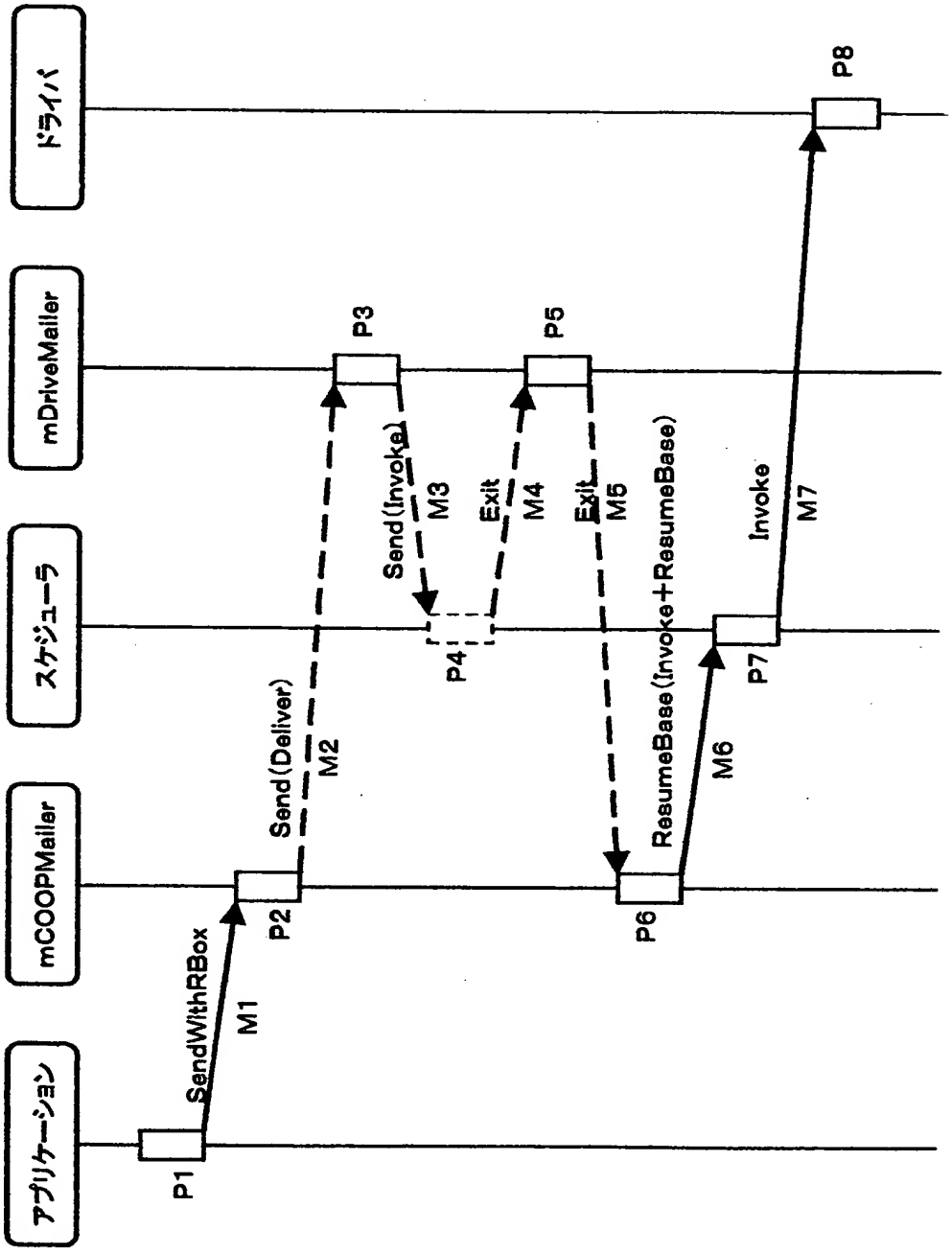
【図８】


開 始
S31
コール･ヒストリの状態
がInternalか？
No
Yes
S32
呼び出し元へ
実行を復帰する
リクエスト･キューの内容
をオブジェクト毎に
1つにまとめて送信する
S33
ExitAPIを発行する
S34
終 了

【図９】


開 始
S41
コール･ヒストリの状態
がInternalか？
Yes
No
S42
スケジューラへの
ResumeBaseメッセージを
リクエスト･キューに格納する
S44
エラーを発行する
S43
リクエスト･キューの内容を
オブジェクト毎に
1つにまとめ送信する
終 了

【図１０】


ドライバ
mDriveMailer
スケジューラ
mCOOPMailer
アプリケーション
P1
P2
P3
P4
P5
P6
P7
P8
SendWithRBox
M1
Send(Deliver)
M2
Send(Invoke)
M3
Exit
M4
Exit
M5
ResumeBase(Invoke+ResumeBase)
M6
Invoke
M7

【書類名】　　要約書

【要約】

【課題】　オブジェクト指向オペレーティング･システムにおけるシステム構成の柔軟性などの優位性を維持したままで、メッセージ通信時のコストを削減する。

【解決手段】　複合オブジェクト内でのメッセージ通信のときは、同一の実行スレッドでを実行して、メッセージ通信履歴を同一実行スレッドからの通信状態に更新する。複合オブジェクト外へのメッセージ送信のときは、メッセージ送信先に応じてメッセージを一旦蓄積する。複合オブジェクト内でのメッセージ処理が終了後にメッセージ通信履歴を参照して、異なる実行スレッドからの通信状態であれば、蓄積されたメッセージを１回のメッセージ通信で送信する。

【選択図】　図４

出 願 人 履 歴 情 報

識別番号　〔０００００２１８５〕

１．変更年月日　１９９０年　８月３０日
［変更理由］　新規登録
住　所　東京都品川区北品川６丁目７番３５号
氏　名　ソニー株式会社